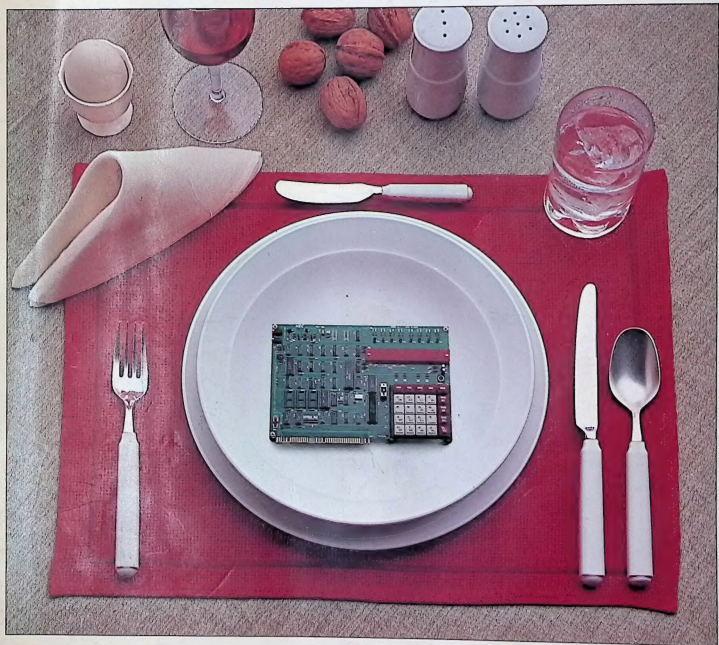


マイクロコンピュータ

TK-85

TRAINING BOOK

ハードウェアとソフトウェア



Nippon Electric Co., Ltd.

TK-8550
MAY, 1982

7-35
TRAINING BOOK

RECEIVED

マイクロコンピュータ
TK-85
TRAINING BOOK
ハードウェアとソフトウェア

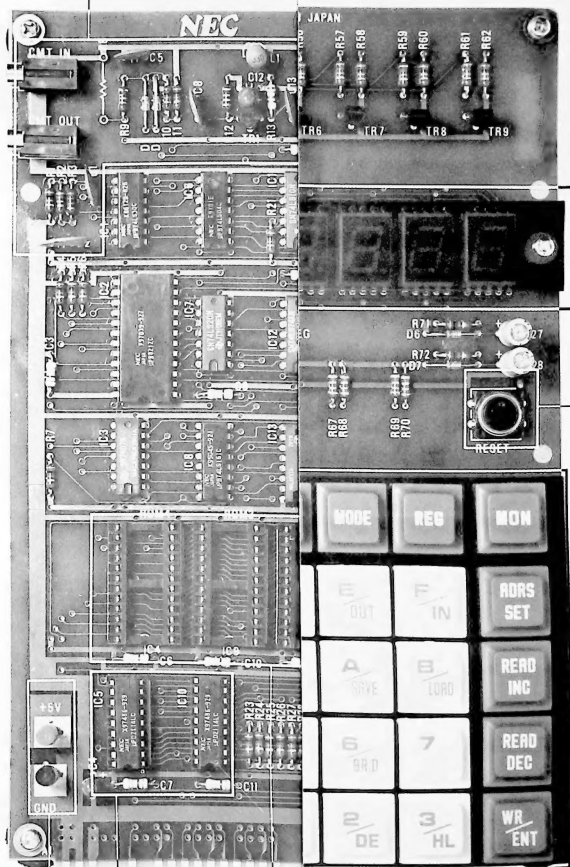
日本電気株式会社

御注意

- (1) 本書の内容の一部または全部を無断で転載することは禁止されています。
- (2) 本書の内容に関しては将来予告なしに変更する事があります。
- (3) 本書は内容について万全を期して作成いたしましたが、万一、御不審な点や、あやまり、記載もれなど、お気づきのことがありましたら、御連絡下さい。
- (4) 運用した結果の影響については、(3)項にかかわらず、責任を負い兼ねますので、御了承下さい。

CMTインターフェイス

リセットスイッチ



電源ターミナル

増設ROMソケット

キーボードスイッチ

は改良のため予告なく変更されることがあります

附属品の確認と足の組み立て

TK-85 の附属品

TK-85 のケース内には TK-85 本体の他に下記に示す附属品が梱包されています。

《附属品》

- TK-85 TRAININK BOOK 1部
- TK-85 本体支持材一式（ゴム足、スペーサ、ビス類）
- 電源用コード 赤黒各1本
- 保証書

以上の附属品があることを確認してください。

TK-85 の足の取り付け方法

TK-85 は足を取り外した状態でケースに梱包されています。使用する前に必ず足を取り付けてください。

ケースには足の取り付け部品として下記のものが附属品袋に入っています。確認してください。

ゴム足	× 7
スペーサ	× 7
ビス (M 3, 25mm)	× 7
ナット (M 3)	× 3
平ワッシャ	× 3

ケースより TK-85 本体を取り出したとき、ケーススイッチ取り付け金具は図 A の1~4 で示す4本のビスにより基板に仮留めされています。
この仮留めしてある4本のビスを外します。

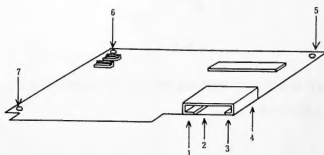
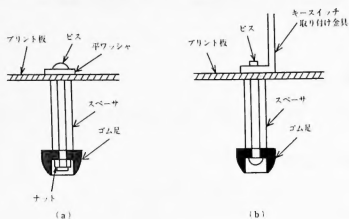


図 A

次に図Aの1~7で示す穴7箇所にて図Bで示す構成どおりに基板へ足を取り付けてください。



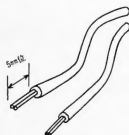
図B 取り付け方法

取り付け方法として図Aの穴1~穴4までは図B(b)の方法で、穴5~穴7までは図B(a)の方法で行ってください。

仮留めビスを外した後の取り付け作業は慎重に行ってください。大きな振動を与えますとキースイッチのリード線が基板より外れる場合があります。注意してください。

電源コードの接続方法

附属品の電源コードを TK-85 本体に接続するには図Cのように電源コードの先端の被覆を取り除きます。



図C

被覆を取り除いた後、露出部分をよじり本体の電源ターミナルに接続します。接続する場合赤色コードを+5Vと決めると便利です。

ま え が き

トレーニング マイクロコンピュータ TK-85は、マイクロコンピュータを理解し、マイクロコンピュータを利用したアプリケーションを開発される方の入門機として最適な製品です。

TK-85は広く一般に知られている当社製品 TK80/80E の流れをくむもので同一の命令で動作する一方、それ以上の機能を含み、従来よりもモニタプログラムが強化され、プログラミングやデバッグが簡単に行え、さらにハードウェアの拡張も簡単に行えるように配慮されています。

本書には、TK-85を御利用頂くための説明のほか、マイクロコンピュータの基礎に関することがわかりやすく解説されています。また TK-85を利用したプログラム例も記載されていますので、実際に TK-85を動作させながら体験的にマイクロコンピュータを学んでいくことができます。

TK-85および本書が広い分野で活用され皆様のお役に立つ事を期待いたします。

目 次

まえがき	(1)
附属品の確認と足の組立て	(7)
TK-85の足の取り付け方法	(7)
電源コードの接続方法	(8)

0章 マイクロコンピュータの魅力

1. はじめに	1
2. マイクロコンピュータの誕生までに	1
3. マイクロコンピュータの利点、欠点	3
4. マイクロコンピュータの応用	4
4.1 自動車への応用	5
4.2 ゲーム機器への応用	6
4.3 パーソナルコンピュータへの応用	7
5. マイクロコンピュータの基礎	8

1章 マイクロコンピュータを動かしてみる

1. はじめに	11
2. TK-85を使う前に	11
3. TK-85を動かしてみる	12
4. テープレコーダとの接続	19

2章 マイクロコンピュータの中をのぞく

1. はじめに	23
2. 2進数と16進数の話	23
3. 8085 CPUの話	28
4. 8085 CPUの命令について	31

3章 TK-85の操作方法

1. はじめに	35
2. キーの説明をする前に	35
3. データ入力	35

4. NORMAL FUNCTION	36
4.1 ADDRESS SET AND MEMORY READ	36
4.2 ADDRESS INCREMENT AND MEMORY READ	37
4.3 ADDRESS DECREMENT AND MEMORY READ	37
4.4 MEMORY WRITE AND ADDRESS INCREMENT	37
4.5 RUN	39
4.6 RESET	39
4.7 MONITOR と CONTINUE	39
5. MODE FUNCTION	40
5.1 IN	40
5.2 OUT	41
5.3 MOVE MEMORY	41
5.4 TEST MEMORY	42
5.5 CASSETTE SAVE	43
5.6 CASSETTE LOAD	44
6. REGISTER FUNCTION	45
7. ステップ動作	45
8. ブレイク動作	46

4章 プログラムの基礎と作り方

1. はじめに	49
2. プログラムとは何か	49
3. プログラムの基礎	50
3.1 フローチャート	50
3.2 コーディングシート	51
4. 8085 CPU の命令体系	53
5. 矢印プログラムについて	57
6. アセンブラ言語について	57

5章 データの取り扱い

1. はじめに	59
2. データの転送	59
2.1 レジスタ間のデータの転送	59
2.2 番地の直接指定によるデータの記憶と引用	60
2.3 番地の間接指定によるデータの記憶	62
2.4 番地の間接指定によるデータの引用	63
3. データの算術演算	64
3.1 加 算	64

3.2	減算	67
3.3	2進化10進数による加算	69
4.	論理演算	70
4.1	否定	70
4.2	論理積	71
4.3	論理和	72
4.4	桁移動	74

6章 制御文

1.	はじめに	77
2.	演算結果の判断	77
3.	実行流れの分岐	80
3.1	無条件分岐	80
3.2	条件付分岐	80
3.3	3方向以上への分岐	82
4.	繰り返し処理	82
5.	データの入出力	84

7章 サブルーチンの考え方

1.	はじめに	89
2.	サブルーチンの定義	89
2.1	無条件リターンのサブルーチン	89
2.2	条件付きリターンのサブルーチン	90
3.	サブルーチンの引用	92
3.1	無条件引用	92
3.2	条件付き引用	94
4.	二つ以上のサブルーチンの引用	96
5.	スタックの利用	97
5.1	サブルーチン実行後の戻り番地の記憶	97
5.2	レジスタ対のデータの退避	100
6.	割り込み処理	100

8章 モニタの詳細な説明

1.	はじめに	103
2.	フローチャート	104
3.	メモリマップ	106
3.1	システムメモリマップ	106

3.2 RAMメモリマップ	107
4. モニタを理解するために	109
5. 分岐とフラグ	109
5.1 モードフラグ (MODE: 83DF番地)	109
5.2 レジスタフラグ (REG: 83DE番地)	110
5.3 トラップフラグ (FTRAP: 83DC番地)	111
5.4 キーフラグ (FKY: 83DD番地)	112
6. 割り込み処理とイニシャライズ	112
6.1 0番地スタート	112
6.2 TRAP割り込み処理	112
6.3 RST 7.5割り込み処理	113
6.4 RST 6.5割り込み処理	114
6.5 リスタートジャンプテーブル	114
7. モードファンクションの説明	114
7.1 INモード, OUTモード	114
7.2 MOVEモード	115
7.3 TEST MEMORYモード	117
7.4 SAVEモード, LOADモード	119
8. μ PD8255のプログラミング	120
9. RIM, SIMについて	123

9章 モニタプログラムリスト

モニタプログラムリスト	125
-------------------	-----

10章 モニタサブルーチンの使い方

1. はじめに	141
2. サブルーチンの考え方	141
3. 表示用サブルーチン	142
3.1 表示用サブルーチン構成	143
3.2 機能説明	143
3.2.1 SEGCV	143
3.2.2 SEGCG	148
3.2.3 RGDSP	150
3.2.4 DLWCH	151
3.2.5 DWCH	153
3.2.6 AWCH	154
3.2.7 CH	155
4. 入力用サブルーチン	157

4.1	機能説明	157
4.1.1	INPUT	157
4.1.2	KEYIN	160
5.	シリアルデータ入出力用サブルーチン	161
5.1	シリアルデータ出力用サブルーチン機能説明	161
5.1.1	REDER	161
5.1.2	SRIOT	162
5.2	シリアルデータ入力用サブルーチン機能説明	165
5.2.1	RDSCH	165
5.2.2	SRIIN	166

11章 TK-85のハードウェア

1.	はじめに	171
2.	TK-85システム構成	171
3.	アドレスバス、データバス	173
4.	ROM, RAMの構成	174
5.	リセット回路	175
6.	トラップ回路	177
7.	表示回路とPMA転送	177
8.	μ PD8255とキーボード回路	180
9.	1インストラクションステップ動作回路	181
10.	CMT回路	183
11.	カードエッジ信号表	184
12.	RAMの増設	185
13.		

付録

1.	μ PD8085ACデータシート	
	特 徴	187
	端子接続 (TOP VIEW)	188
	ブロック図	188
	端子機能	189
	割込みとシリアルI/O	191
2.	TK-85全回路図	
	8085CPU回路図	193
	メモリ回路図	194
	LED, キーボード回路図	195
	CMT回路図	196
3.	TK-85部品配置図	196

0 章 マイクロコンピュータの魅力

1. はじめに

本章ではまず初めにマイクロコンピュータがどのようにして誕生したかを述べ、次にそのマイクロコンピュータがいかに利用されているかを見ていきます。そして最後に簡単にマイクロコンピュータの概念について勉強することにしましょう。

2. マイクロコンピュータの誕生まで

1948年、この年に世界最初のコンピュータ ENIAC がアメリカのペンシルバニア大学で誕生しました。この ENIAC は、大きさが二十畳程の部屋一杯を占めるくらいの巨大なもので、真空管やリレーを何万個も使用しており、数百キロワットの電力を消費するものでした。

現在のコンピュータと比べるとその性能はきわめて低いものでしたが、それでも当時の人達にとっては、機械が自動的に計算をしたり、いろいろな処理をすることができるということは大きな驚きでした。

このようにして生まれたコンピュータはその後、半導体技術と相まって進歩を重ね、今日に至っています。現在では、コンピュータの性能は非常に高く、そのため、コンピュータといえば神様のような万能の機械と思っている人さえいるほどです。

それでは、なぜコンピュータはこれほどまで進歩し続けてきたのでしょうか？ コンピュータはどのような点で人間社会に貢献してきたのでしょうか？

それはコンピュータには人間の能力より優れている点がいくつかあったからです。その人間より優れている点として次の三つの事柄をあげることができます。

- (1) 計算・処理速度が人間に比べて桁違いに速い。
- (2) 単純な繰り返しあるいは複雑な作業を長時間やらせても疲れないし、またミスもしない。
- (3) 一度記憶したことは絶対に忘れない。

これらのコンピュータの長所は、人間が数多くの情報を処理したり、複雑な計算をしたり、たくさんのことを記憶しておいたりするには大変都合のよいものだったのです。大量の情報を処理したり、複雑な計算を人間がやっていたのでは時間がかかるし、ミスをするかもしれません。また、記憶しておくべきことを忘れてしまうこともあります。

コンピュータを使用すればこれらの仕事を正確に速くやってくれます。このコンピュータの長所をより強力に生かすために、コンピュータは進歩し続けてきたのです。

さてこのように進歩してきたコンピュータの歴史の中で、我々がこれから見ていこうとするマイクロコンピュータはどのようにして生まれてきたのでしょうか。

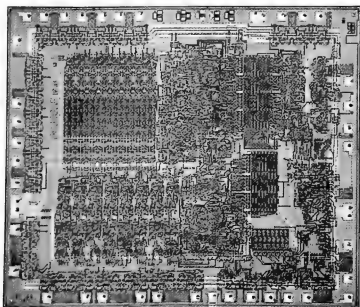
エレクトロニクスの進歩は目を見張るものがあります。より高性能かつ小型で低消費電力のものを求めて真空管からトランジスタへ、そしてトランジスタからIC(集積回路)へ、さらにはLSI(大規模集積回路)へと飛躍的に進歩が続いています。現在の最新鋭のLSIはわずか数ミリメートル四方のシリコン基板上にトランジスタの数にして数万個を詰め込んだものが作られ、高性能・小型・低消費電力、そしてある程度数の作れば非常に安価なものが実現されています。

このようなLSIは多くの分野で使用され、現在ではLSIはエレクトロニクスの大きな柱となっています。しかし、各分野で使うLSIはそれぞれ全く違った仕様が要求されます。ですから各分野からの要求に応えるためにはそれぞれ違った仕様のLSIを何種類も作らなくてはなりません。しかし、LSIを作るのは非常に面倒で、一つのLSIを新しく作るには長い時間と多くの人手がかかります。しかし、LSIはひとたび作りだすと、同じ仕様のものを比較的簡単に大量生産することができるといふ長所があります。そこで次のような考え方が生まれました。

「利用者から注文があるたびに新しいLSIを作るのではなくて、どうにかしてどんな分野にも使えるLSIを作ることはできないのか？」

これに対する解決策はLSIにコンピュータの機能をもたせることでした。コンピュータはプログラムを変えれば、どのようにも利用できることからLSIにコンピュータの機能をもたせ、あとはプログラムを変えるだけで、同一LSIであらゆる分野の要求に応じようという考えです。

つまりそれまでのLSIを印刷された紙と考えるなら、コンピュータの機能を持たせたLSIは真白の紙であり、利用者がそれぞれ自由に書きこめばよいようにしたのです。



LSIの内部拡大写真 (μ PD8080AF)

3. マイクロコンピュータの利点、欠点

もうおわかりになったでしょう。そうです、コンピュータ化された LSI、または LSI 化されたコンピュータといってもよいのですが、それがマイクロコンピュータなのです。

このアイデアを実現したものが1971年、アメリカの INTEL 社より発表された4004というマイクロプロセッサで、今日のマイクロコンピュータブームを生み出すもととなりました。

3. マイクロコンピュータの利点、欠点

2節で述べたようにして生まれたマイクロコンピュータはそれまでのコンピュータの利点と LSI の利点の両方をかねあわせて持っています。つまりコンピュータの長所である処理速度が速く、正確でミスをしないなどに加えて、コンピュータが小さく、軽く、安くなったのです。

世界最初のコンピュータ ENIAC 以後、コンピュータの性能は大きく進歩していき、そして多くの分野の人々に使われてきましたが、絶対数からすれば非常に少数の人々にしか利用できませんでした。なぜでしょうか？ それは、コンピュータは高価なものであり、専門の人以外には容易に利用できるものではなく、形も大きなものだったからです。ですからコンピュータを利用したくても価格の点や大きさの点などで使用することのできない人が多かったのです。

ところが、そこへマイクロコンピュータが登場したのです。それまでのコンピュータに比べて価格は大変安く、小さく、また性能のほうもそれまでのコンピュータに比べてそう劣ってはいないのです。これが世間の注目をひかないわけがありません。

実際、マイクロコンピュータはたちまちのうちに多くの人々の、それも今度は絶対数にしても多くの人々から興味をもたれたのです。



マイクロコンピュータCPU

本節の題に「マイクロコンピュータの利点・欠点」と書きましたが、マイクロコンピュータは大容量・超高速処理の要求を除けば、それまでのコンピュータに見られる多くの使用上の制約がそのまま利点となり、利点は数多いけれども欠点はなかなか見つからないほど素晴らしいものになっているのです。

今まで、コンピュータの使用を考えたときにあった障害のほとんどを取り除いたのですから、これからはいろいろな分野にマイクロコンピュータの使用を考えることができるのです。その応用範囲は無限といっていいぐらい広がったといえます。もうコンピュータは特殊なものではなくなったのです。

これはある一部の人達にしか使用できなかったコンピュータがあらゆる分野の人達にも使えるように道が開けたという点でコンピュータの歴史の中で的一大エポックなのです。

読者のみなさんもマイクロコンピュータの知識を得られたなら、マイクロコンピュータを活用する新しい分野をみつけ、そしてそれを開拓していただきたいと思います。

4. マイクロコンピュータの応用

マイクロコンピュータは安価で小型軽量、低消費電力など数々の長所を生かしていろいろな分野に活用することができると書きましたが、具体的には今後どのような使われかたをしていくのかを考えてみましょう。

文明の進歩は我々の生活を次第に豊かにしていきました。我々は機械を使うことを覚え、その機械を使う便利さを得てきました。

しかしこのような社会において、我々は機械を操ってはいませんが、逆に機械に操られていることも少なくないのです。そのような悲劇をあつかった映画にチャップリンの「モダン・タイムス」がありましたが、我々は機械を100%操ってはいないのです。

たとえば自動車について考えてみましょう。

我々が本当に自動車を奴隷のように100%操っているのなら、自動車に向かって「オイ、駅へ行け!!」などと命令すれば自動車が自動的に命令されたとおりに駅へ行くのが本当ではないでしょうか？

現実はどうでしょう。自動車を動かすにあたって「ガソリンははいっているか？」とか、運転しているときには「事故をおこさないようにしなければ」などと思いながらハンドルを動かし、目を常に前後左右にひからせているではありませんか！

勿論、なかには「そのように運転するのが楽しいんだ」という人もいるかもしれませんが、それは別として、一般に我々は自動車を奴隷のようにあつかっていないことがわかるでしょう。

それでは自動車の「行け!!」と言っただけで自動車が進むにはどうすればよいのでしょうか。

いちばん良いのが自動車自身にロボット、つまり機械人間になってもらい、自動車が自分で「ガソリンはあるか」とか「左右に人はいないか」などを判断して動くようになればいいのです。

少し話が現実味をもたないようですが、たとえばロケットはコンピュータによって操縦されていることや、飛行機なども離陸から着陸までコンピュータにまかせっきりにできる現実を考えるなら、そう現実味のない話ではありません。

話をもっと身近なものにしましょう。

あなたの家にクーラーがあるとします。あなたはそのクーラーを使用するにあたって「何時になったらクーラーを作動させ、室温が〇〇度になったら、その後その温度を常に一定に保つように動作する」ようにしたかったとします。

もし、クーラーが機械人間になってくれたなら……。今度の話はもう実用化されているものです。

ではこの機械人間はどのようにすれば可能になるのでしょうか。そうです。このような場合にマイクロコンピュータを使用するのです。

今までのコンピュータを使用しても機械人間を作ることは可能ですが、コストの点や、クーラーより大きなコンピュータを別に設置しなければならないなど非現実的なものとなります。マイクロコンピュータは小さいものですから、クーラーの中に組み込めます。また、価格が安いからクーラー自身の価格にもそうひびきません。

マイクロコンピュータはこのようにいろいろな機器に頭脳を持たせて、人間の仕事を肩代わりさせるための強力な武器となります。

もちろん今までコンピュータで行っていた仕事の一部をマイクロコンピュータに肩代りさせることもできます。しかし、マイクロコンピュータは、マイクロコンピュータにしかできない応用に用いたとき、その効果が如何なく発揮されます。その一つの例が今まで述べてきたいろいろな機械を、外観はほとんど変化させずに機械人間にすることなのです。

近い将来、あらゆる機器が機械人間になり、人間がどんな機械も100%操れるようになるのではないのでしょうか。

ではこのような考え方にそって、実際、マイクロコンピュータは現在どのように活用されているのかを紹介しましょう。

4.1 自動車への応用

先程自動車の究極的なかたちまでを考えましたが、現実はまだまだ夢物語です。逆にいえば、自動車という機械は人間に大変多くの動作を要求してくる機械なのです。ですから、自動車にはコンピュータを導入する部分がたくさんあるのです。

現在、自動車にマイクロコンピュータが導入されている部分は主にエンジン関係やブレーキ関係などです。

近年、深刻な石油不足から、自動車に対して、低燃費、つまり、より少ないガソリンでより長い距離を走る自動車が求められています。

このような状況下において、各自動車メーカーはエンジンをいついかなるときにも最も効率よ

く働かせるように努力しています。しかし、エンジンの機械的部分そのものは飛行機の技術などとあいまってきわめて高い水準に達してしまっているため、機械的部分の改良の余地はほとんどないくらいになっています。そこで、逆にエンジンをいかなる状態でも最も効率よく働くようにコントロールすることを考えました。

たとえば、エンジンの最適点火時期を決定させること。あるいはエンジンが最も良く働くことされる空気・燃料比（14.7対1といわれる）を常に保つために吸入空気量と排ガス中の酸素の比を測定して、燃料噴射装置をコントロールするなどです。

ブレーキ関係では、走行中の自動車が停止するには、ブレーキペダルを踏むことによって油圧を上昇させそれがホイールに伝わり、その結果タイヤと路面の摩擦によって止まります。そのとき、あまり強くブレーキペダルを踏むとホイールがロックされてスリップしてしまい、制動距離がかえって長くなってしまったり、路面の状態次第では、操縦不能になることさえあります。ブレーキがいちばん良くきくのはホイールがロックされる一歩手前の状態を保つときです。これを実現する装置をアンチスキッドブレーキといいます。これはホイールのロックを検出したら油圧を自動的に下げ、ホイールのロックを解除する機構になっています。

以上、エンジンとブレーキの二つの例をあげましたが、この二つとも制御部分にそれぞれマイクロコンピュータが使われています。これらの部分はマイクロコンピュータが出現しなければ全く不可能か、または大変困難なことだったのです。

このようにして自動車は現在、着々と完全な機械人間に向けて進歩しています。近い将来、完全な機械人間となった自動車ができることでしょう。

4.2 ゲーム機器への応用

インベーダーブームというのがありました。御存知だと思いますが、あのテレビゲームのインベーダーゲームのことです。みなさんの中にも目を血走しらせながら100円玉を湯水のごとく使って、インベーダーゲームにしがみついていた時期があったかもしれません。このインベーダーゲームをはじめ、最近に見られるテレビゲームの類のほとんどにマイクロコンピュータが使用されています。この応用例は我々の生活を豊かにする機器の自動化・省力化とはちょっと異なる点がありますが、我々に従来と違った娯楽をあたえてくれるという応用では大変身近なものですから、マイクロコンピュータの応用例として少し書いてみましょう。

ブロックくずしというゲームを御存知ですか？ そうです、「ビポッ・ビポッ」といいながら落ちてくる玉を下方にあるパドルではね返し、上方にあるブロックを消していくゲームです。さて、このブロックくずしというゲームは、画面などを見ればわかるようにインベーダーゲームに比べて、比較的単純な処理しか必要としません。しかし、このブロックくずしでさえ、もしLSIを使用せずに作るとなるとその電子回路は大変複雑なものになり、その結果、人手と時間がかかり、コストアップを招き実用化（つまりゲームなどにはそうたくさんのお金はかけられないから）はむずかしいといえます。

だからインベーダーゲームをはじめとするテレビゲームなどではLSIを使わずに作ることは非現実的なことといえます。そのLSIとして、マイクロコンピュータが使用されているのです。もし普通のLSIをつかうと、新しいゲームを開発するたびに新しいLSIを開発しなければなりません。

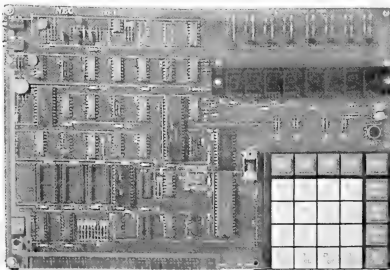
マイクロコンピュータには、コンピュータの特徴であるプログラム次第でいろいろな働きをするという長所があります。このことはプログラムを開発することでいろいろな種類のLSIを作り出すことができるといえます。しかもプログラムの開発は人間の頭脳によるだけで多大な設備はいりません。ですから、マイクロコンピュータの知識をしっかりと持っていれば、新しいゲームの開発が大変らくになりますし、また自分でいろいろなゲームを作り出すこともできます。

現在、このようにして新しいゲーム機器が次々と開発され、ゲーム場に送り込まれています。あなたがそのゲーム機器でゲームを楽しんでいるとき、それはとりもなおさず、あなたはマイクロコンピュータを操っているのです。

4.3 パーソナルコンピュータへの応用

今までのコンピュータは高価で形も大きく、大変あつかいづらいなどから、個人が自分専用のコンピュータを所有することは本当に夢物語でした。

そこへマイクロコンピュータが登場しました。マイクロコンピュータは、それまでのコンピュータでは考えられなかった応用が開けてきたのと同時に、それまでのコンピュータの性能をある程度保ったまま大変安価なコンピュータを供給できるようになったのです。そしてその値段は個人で買える値段のものさえできました。そこで、個人専用のコンピュータという考え方、すなわちパーソナルコンピュータという考え方ができたのです。



TK-85外観

さて、このコンピュータを個人所有していったいなにをするのでしょうか？

自分だけの情報、記憶、管理すべきことをこのパーソナルコンピュータに肩代りしてもらうこともできるでしょう。

教育機械としての応用、つまりコンピュータをうまくプログラムすることによって教科書の代りや先生の代りをさせることもできるでしょう。あるいは趣味としている人もいるかもしれません。

また、パーソナルコンピュータを使ってマイクロコンピュータの勉強をするということもできます。

このパーソナルコンピュータはマイクロコンピュータの応用分野として新しく開拓されたので、今後、個人用として、また家庭内において、いろいろな利用方法が考えられます。

以上三つのマイクロコンピュータの応用例を見てきましたが、マイクロコンピュータの応用例はこれだけにとどまっていません。ちょっと思いうかべただけでも、カメラ、ミシン、オーディオ機器、クーラー、テレビ、計測器、印刷システム、時計、シンセサイザ、教育用機器、アニメーション、おもちゃ、自動販売機、工場における制御機器、大型コンピュータの端末…… etc. とあります。

このように、マイクロコンピュータの応用分野は無限に広がっていき、もはやコンピュータはひとつのパーツとして扱われる時代を迎えています。

5. マイコンピュータの基礎

今まで、マイクロコンピュータの生い立ちやその応用についてお話ししてきました。次章からマイクロコンピュータを実際に使って勉強していただくわけですが、その前にマイクロコンピュータを勉強していく上での基礎知識としてマイクロコンピュータの概念を述べておきます。

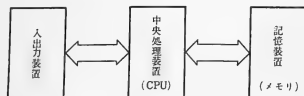


図0.1

図0.1を見て下さい。マイクロコンピュータは大きくわけて三つのブロックに分かれています。それぞれの三つのブロックを簡単に説明してみましょう。

- ① 入出力装置とはマイクロコンピュータの外部からの信号をマイクロコンピュータ内部へ受

け入れる役割をもっています。また逆にマイクロコンピュータの内部の信号を外部に outputs 役割もします。人間でいうと目、耳や口にそれぞれあたります。

② 中央処理装置とは入力装置から入ってきた信号や記憶装置内にある内容によって実際にいろいろな仕事をするところであり、コンピュータの中心をなすところです。

③ 記憶装置はその名のとおりに、いろいろな情報を記憶するところです。その情報は中央処理装置の命令によって記憶されることもあれば、その記憶した内容が呼びだされることもあります。

中央処理装置と記憶装置は人間でいえば頭にあたるでしょう。

なお中央処理装置のことを英語で CPU (Central Processing Unit の略です) といい、この言葉が使われることが多いので、以後中央処理装置のことを CPU と呼ぶことにします。同様に記憶装置のことはメモリ (Memory) と呼ぶことにします。

マイクロコンピュータは以上述べた部分から構成されています。しかし、マイクロコンピュータは電源を入れただけでは、動作しません。マイクロコンピュータは、最初なをすればよいのか知らないのです。それでは最初どのようにして動作させるのでしょうか。

マイクロコンピュータにかぎらずコンピュータと名のつくものは次の二つの条件をもっています。

- (1) メモリの中に処理の手順を書いたプログラムを持っていること。
- (2) 自動的にそれらの処理を実行することができること。

上記の二つの条件はどのような意味なのでしょう。

先程 CPU はいろいろの処理を実行する部分であるといいましたが、CPU それ自体でできることはごく基本的なことだけなのです。それにもかかわらず複雑で難しい処理をコンピュータができるのは、ごく基本的な命令を巧みに組み合わせておくことによってコンピュータは見かけ上複雑で難しい処理を行っているのです。

今までプログラムという言葉は何回に使ってきましたが、この基本的な命令を巧みに組み合わせ、コンピュータを見かけ上複雑な処理ができるようにすることをプログラムするといいます。

このようにコンピュータは、プログラムを与えることによって始めて動作します。そして、このプログラムを一度実行させたら、あとはコンピュータはプログラムに書いてあるとおりの手順を自動的に実行していくのです。

さらにコンピュータを詳しく見ていってみましょう。CPU はどのようにしてメモリからプログラムを読み出し、命令の実行をしていくのでしょうか。

プログラムというものはプログラムが書かれた段階ではそれ全体としてどんな仕事をするにせよ、ある基本的な命令が順番にならんでいます。

そのようになっているプログラムをメモリに記憶させておいて、それぞれの命令を CPU がメモリから呼びだし CPU が実行するのです。そのとき CPU が、それぞれの命令の順番を順番を知るめやすとして、それぞれの命令にプログラムに書いてある順番に番地 (Address: アド

レス)をつけておくのです。たとえばメモリの1番地にある命令、2番地に次の命令、以下3番地、4番地、……というように続いていくのです。そしてCPUはプログラムの実行時にはその番地を順に読んで、そこに書かれている命令を順に実行するという方法をとっているのです。

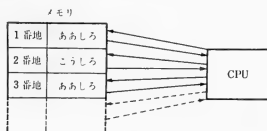


図0.2

メモリの基本的な働きはある情報を自由に記憶したり、あるいは思いだすことです。メモリに情報を記憶させることを書き込み (Write)、思いだすことを読み出し (Read) といいます。さてマイクロコンピュータに使われているメモリにはどのようなものがあるでしょうか？

マイクロコンピュータに使われているメモリは大きくわけて二つあります。その一つをRAM (Random Access Memory: ラム) といいます。マイクロコンピュータに使われているRAMは半導体回路でできているのですが、普通、半導体で作られたRAMは電源を切ると記憶内容はメチャメチャになってしまいます。しかしRAMは読み出し、書き込みとも自由にできる便利なメモリです。他の一つにROM (Read Only Memory: ロム) というメモリがあります。これはRAMと違い電源を切ってもその内容は保存されますが、読み出しだけにしか使えないメモリです。RAMもROMも別にどちらが良いというわけではなくどちらも一長一短があり、場合に応じて使いわけする必要があります。

この章ではマイクロコンピュータの生いたちからその応用例、そしてマイクロコンピュータの基礎についてお話してきました。では次章から、実際にマイクロコンピュータの世界に足をふみ入れていきましょう。

1章 マイクロコンピュータを動かしてみる

1. はじめに

0章のおわりでマイクロコンピュータの基礎知識についてお話しましたが、まだほかにも知らなければならないこともたくさんあります。しかし習うより慣れろという諺もあります。詳しい知識の説明をするまえに TK-85 を使って、とにかく自分の手でマイクロコンピュータを動かしてみましょう。

2. TK-85 を使う前に

まず TK-85 を使う前にそれを動くようにセットしなければなりません。

TK-85 を動かすには電源として + 5 V の電源を用意し、TK-85 についている + 5 V 端子と電源の + 5 V 端子、および TK-85 の GND (アース) 端子と電源の GND 端子を、それぞれ間違いないようにしっかりとつないで下さい (図 1.1 参照)。

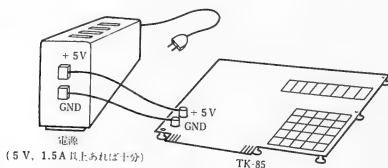



図 1.1

次に電源を投入する前に TK-85 の各部分の簡単な説明をしておきます。TK-85 の右下のほうに 25 個の押しボタン (キー) が並んでいます。この部分をキーボードといい、キーを押すことにより TK-85 にプログラムを入力したりいろいろな命令を TK-85 に入力します。このキーボードの上に数字などを表示する赤く点燈する部分があります。この部分のことを LED ディスプレイ (以下、単に LED と記します) といいます。この LED が人間に対する表示機能です。これら、キーボードおよび LED が 0 章で述べたコンピュータの入出力装置に当たります。そしてほかのいろいろな半導体や抵抗、コンデンサなどが配置されているところが CPU およびメモリの部分と考えてください。

3. TK-85 を動かしてみる

それでは、電源を入れましょう。電源を入れたら次に  と書かれたボタンを押します。LED は次のように表示するはずですが、



もしこのように表示されなかったり、電源を入れたあと TK-85 から煙やへんな臭いなどがしたらすぐに電源を切って TK-85 と電源との配線に間違いがないかを見て下さい。

LED が上記のようになったら TK-85 は正常に動いています。

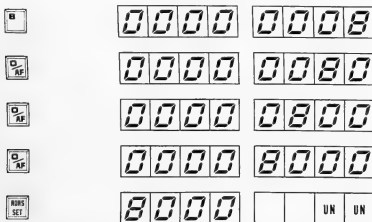
つづけて次の順にキーを押して試してみてください。



以下に上記のように押したときの LED の表示を示して見ます。

押すキー

押し終わった後の LED 表示



上記の最後の行にある UN という表示はなんらかの文字で表示されますが、それが一定していません。UN としてある箇所は何か文字あるいは数字が表示されれば、どんな表示であっても気にすることはありません。

先を続けましょう。いままでの操作で LED は次のような表示になっています。



つづけて、次の順にキーを押します。



以下にそのときの LED の表示を示します。

押すキー

押したあとの LED の表示



いまやっていることを説明しておきます。この LED の左 4 桁の表示はメモリの番地を示しています。LED の右の 4 桁はそのメモリの番地に書き込まれる内容です。つまり今、メモリの 8000 番地に 21 というデータ（情報）を書き込んだのです。

ただし、この 8000 番地の 8000 やデータの 21 は我々の普段使う 10 進数の 8000 や 21 ではなく、16 進数の 8000 や 21 です。そのため、0～9 のほかに A b C d E F などという文字が 0～9 の数字にまじっていることがあります。16 進数の話は 2 章でしますので、とにかくこの 8000 や 21 は我々が普段使用している 10 進数の数字とは少し違っていることを覚えておいて下さい。

ここでキーを押し間違えたときのことを説明しておきます。たとえば と押すべきところを と押ししてしまったら、 を押す前に続けて と正しく押し直した後 を押せばいいのです。こうすれば間違わずにキーを押したときと同じように TK-85 は理解してくれます。

この過程を LED の表示とともに示しておきます。

キーを押す順序

そのときの LED の表示



（間違えた）



（押し直し）



（正しく押せたことになる）



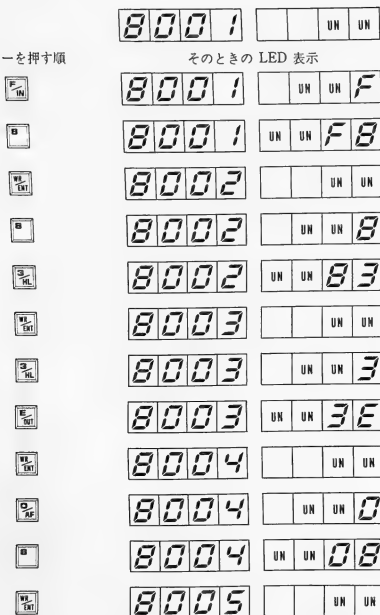
また全然ちがうキーを押してしまったら、 ボタンを押して最初からやり直して下さい。

話を続けます。次のようにキーを押して行って下さい。いままでの操作で LED は下のように表示しています。

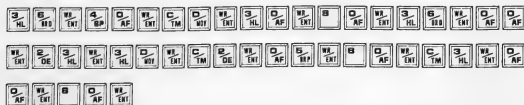
1章 マイクロコンピュータを動かしてみる

キーを押す順

そのときの LED 表示



以下、LED の表示は略してキーを押す順番だけを書きます。キーを押し間違えたときは前述のように押し直して下さい。



ここで と押します。LED の表示は

8030 UN UN

となります。そして次の順にキーを押し続けます。



以上でキー入力は一応終了ですが、もしかしたら気がつかないうちに押し間違いをしているかもしれません。そこでチェックをしてみます。

以下、正しく押されたときの状態をお見せしますから、あなたの TK-85 の LED の表示と比べてみて、もし表示と違っていたら直します。その直しかたはあとで説明しますから、とにかくあなたの LED の表示と比べて、違ったところがあったらその箇所をなにかにメモしておいて下さい。

と押します。

以下、押したキーと LED の表示は次のようになります。

押したキー

LED 表示

	8000	21
	8001	F8
	8002	83
	8003	3E
	8004	08
	8005	36
	8006	40
	8007	Cd
	8008	30
	8009	80







1章 マイクロコンピュータを動かしてみる

READ INC	800A			36
READ INC	800b			00
READ INC	800C			23
READ INC	800d			3d
READ INC	800E			C2
READ INC	800F			05
READ INC	8010			80
READ INC	8011			C3
READ INC	8012			00
READ INC	8013			80

次に     と押します。

READ SET	8030			16
READ INC	8031			40
READ INC	8032			06
READ INC	8033			00
READ INC	8034			05
READ INC	8035			C2
READ INC	8036			34

3. TK-85を動かしてみる

	8037			80
	8038			15
	8039			C2
	803A			32
	803b			80
	803C			C9

さて、以上を見てきて、間違いがひとつもなければ結構です。しかし、もし間違いがあったら次のようにして直しておいて下さい。

間違っていたところはなにかにメモしてあるはずですね。たとえば、

8011			C3
------	--	--	----




と表示されるところが

8011			C2
------	--	--	----

と表示されたとします。

     と押しますと LED に

8011			C2
------	--	--	----

と表示されます。続けて    と押します。これで間違いは直りました。以下にそのときのようなすを示します。

押すキー

LED 表示

現在

8011			C2
------	--	--	----



8011		C2C
------	--	-----



8011	C2C3
------	------



8012		00
------	--	----

1章 マイクロコンピュータを動かしてみる

これで



を



に変えることができました。もし間違った箇所があったらこの例に従って直して下さい。

さあ、これでチェックもおわりました。

もうわかっている人もいるかもしれませんが、いままでやってきたことは TK-85 のメモリにプログラムを書き込んでいたのです。そしてそのプログラムをメモリに書き込む仕事はおわりました。

では TK-85 にプログラムを実行させてみましょう。

次の順にキーを押します。



これで TK-85 がプログラムを実行します。

いかがですか？ LED の上を左から右へ矢印が次々に動いていったと思います。もしそうならなかったら先程のチェックが完全ではなかったのです。もう一度、前の手順に従ってチェックを行って下さい。

なかなか面白いプログラムでしょう。しかしいつまでも見ているとあきてきます。この矢印プログラムの実行をやめるには **MON** を押して下さい。プログラムの実行を止めたときのアドレスとアキュムレータとフラグの内容が LED に表示されます。その後また動かしたかったら **CONT** を押せばよいのです。

この矢印プログラムが、どのようにしてコンピュータによって実行されているかは次章以後で詳しく説明します。ここではこの矢印プログラムとはどのように書かれているかをちょっとお見せしておきます。

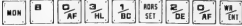
アドレス	ニーモニックコード	オペランド	マシンコード
8000	LXI	H, 83F8H	21F883
8003	MVI	A, 08	3E08
8005	MVI	M, 40H	3640
8007	CALL	8030H	CD3080
800A	MVI	M, 00	3600
800C	INX	H	23
800D	DCR	A	3D
800E	JNZ	8005H	C20580
8011	JMP	8000H	C30080
8030	MVI	D, 40H	1640
8032	MVI	B, 00	0600
8034	DCR	B	05
8035	JNZ	8034H	C23480
8038	DCR	D	15
8039	JNZ	8032H	C23280
803C	RET		C9

リスト 1.1

これを見てもおそらくチンプンカンプンでしょう。でも心配することはありません。本書を読んでいくうちに、すらすらと理解できるようになりますから。

次にこのプログラムをちょっと改造してみます。

次のようにキーを押します。



押し間違えたならもう一度最初からキーを押して下さい。これでプログラムの改造はおわりました。どのように改造されたのか、実行させてみましょう。

先程の手順と同じく、です。

矢印のスピードが速くなったでしょう。今プログラムを改造するときに押したの部分を変える二つの数字に変えてみるとまた矢印のスピードが変わります。試してみてください。

もうひとつやってみます。



またプログラムを改造しました。実行させてみましょう。矢印に代って8が左から右へ移動するようになったでしょう。プログラムはこのように少し変えるだけでずいぶん違った結果がでてくでしょう。

このことは覚えておして下さい。

4. テープレコーダとの接続

これまで TK-85 にプログラムを書き、それを実行してみたわけですが、TK-85 の場合、電源を切るとせっかく入れたプログラムをすべて忘れてしまいます。ですから一度電源を切ってしまったからもう一度矢印プログラムを実行してみたいと思ったら、またこの章の最初からやってきたことを繰り返さなければならないこととなり、大変不便です。TK-85 ではこのような不便をなくするために入力したプログラムをテープレコーダに記憶させておくことができるようになっています。

では実際にテープレコーダにプログラムを保存させてみましょう。

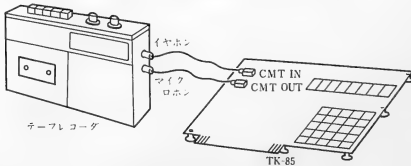


図 1.2

1章 マイクロコンピュータを動かしてみる

まずあなたの手近にあるテープレコーダを用意してください。それと TK-85 を接続します。

TK-85 の IN と書いてあるジャックとテープレコーダのイヤホンジャックを、そして TK-85 の OUT と書いてあるジャックとテープレコーダのマイクロホン入力ジャックをそれぞれつなぎます (図1.2参照)。

これで TK-85 とテープレコーダの接続は完了です。テープレコーダにテープを入れます。

テープを完全に巻き戻し、テープのリーダ部分が巻き込まれるまでテープを少し巻いておきます。

それから、次のようにキーを押します。



すると LED は次のように表示します。



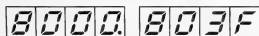
とキーを押します。

LED は再び



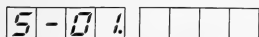
となります。

それから と入力すると LED に



と表示されます。テープレコーダを録音状態にしてスタートさせます。そうしてから、

を押すと以後 LED は以下のように変化します。



最後の状態になったらテープレコーダにプログラムの書き込みは完了です。テープレコーダを止めてテープを巻き戻しましょう。

TK-85 の電源を切りましょう。電源を切って10秒程たったらまた電源を入れます。

RESET を押して下さい。LED は




になっていますね。

では TK-85 は電源を切るとプログラムを忘れてしまうことの証明に、前述のプログラムが入っていた状態と同じ操作をしてみましょう。



どうです、矢印が左から右へ流れましたか？ 流れなかったでしょう。TK-85 は電源を切るとプログラムを忘れてしまうのです。また矢印プログラムを TK-85 に実行させたい場合は、またこの章の始めからの手順でやり直しますか？ そんなことをしなくても大丈夫です。このために TK-85 とテープレコーダを接続して矢印プログラムをテープに記録したのです。ではさっそくテープレコーダから TK-85 にプログラムを入力してみましょう。

テープレコーダの、ボリュームつまみを最大と最小の中間ぐらいにおきます。記録したテープを完全に巻き戻しておいて  と押します。LEDは、

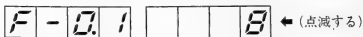


となります。

 と押すと LED は



となります。テープレコーダをスタートさせます。そのうち LED が



となったらプログラムはうまく入力されました。





と押せば矢印プログラムが動くでしょう。

問題は LED が、



となったときです。これはテープレコーダから TK-85 へプログラムがうまく入力できないことを表します。

このように表示されたらテープを巻き戻して、 と押してテープレコーダのボリュームの位置を適当に変えて  を押しプログラムのロードを2～3回試みて下さい。プログラムを TK-85 にうまく入力できたなら、今後、TK-85 と共に使うときは、そのテープレコ

1章 マイクロコンピュータを動かしてみる

ードのボリュームの位置をそのうまくいったときの位置にして使ってください。

2～3回試みて、うまく入力されないときは手数がかかりますがこの章の最初からの手順でやり直して下さい。テープレコーダの故障か接続ミスがないかぎり TK-85 にプログラムがうまく入力できないことはほとんどありません。しかし、何度やってもだめなときは TK-85 を買ったお店などに相談してみてください。

この章では TK-85 をとにかく動かしてみるということを目標にしてきました。あなたの TK-85 はうまく動きましたか？

しかし、あなたが TK-85 を動かせたといってもその原理を十分理解して動かしたではありません。次章からは TK-85 はどうやって動いているのか、その中をのぞいてみましょう。

2章 マイクロコンピュータの中をのぞく

1. はじめに

マイクロコンピュータを自由自在に使用するために知らなければならないことがたくさんあります。その知識は単に電気・電子工学の知識だけではなく、数学の知識なども必要です。

この章ではこれらの知識の中から特に重要な知識をセレクトして、それらをやさしく述べてみることにします。

2. 2進数と16進数の話

コンピュータは2進数しか扱うことができないということを聞いたことはありませんか？ そのとおりなのです。それはマイクロコンピュータでも同様です。では2進数とはいったいどのような数でしょう。

我々は日常、10進法で表される10進数を用いています。しかし、日常使われている記数法には10進法以外のものもあります。たとえば、12個で1ダース、12ダースで1グロス(12進法)、また60秒で1分、60分で1時間(60進法)などがあります。

これらはすべてその桁上がりに違いがあります。2で桁が上がる数、それが2進数なのです。2進数は2で桁が上がるので

0, 1

の二つの文字しか必要ありません。このことは10進数に比べ計算が非常に簡単になります。

マイクロコンピュータの電気回路では、2進数を表すために異なった二つの電圧で実現しています。たとえば電圧5Vを1(信号がある)、0Vを0(信号がない)というように対応させているのです。このように異なった二つの電圧の信号のみで2進数が表現できるということは、コンピュータの回路構成が簡単になります。もし10進数でコンピュータを実現すると異なった電圧が10種類も必要となり、回路構成が複雑になり信頼性も低下します。

それではここで2進数について、その計算方法を具体的に説明しておきましょう。

2進数の足し算に使われる法則は次の三つだけです。

$$0 + 0 = 0$$

$$0 + 1 = 1 + 0 = 1$$

$$1 + 1 = 10$$

注意しておきますが最後の10は「ジュウ」と呼ばずに「イチ、ゼロ」と読みます。すなわち

10のはじめの1は桁が上がったことを示します。10進数の10と2進数の10は違うことに気をつけて下さい。またこれ以後10進数と2進数を区別するために、2進数の表現には 10011_2 のようにならに小さく $_2$ と書くことにします。

2進数の掛け算の法則も次の三つです。

$$0_2 \times 0_2 = 0_2$$

$$0_2 \times 1_2 = 1_2 \times 0_2 = 0_2$$

$$1_2 \times 1_2 = 1_2$$

2進数の引き算、割り算の法則はどんなものでしょうか？ 上の例を参考にして考えてみて下さい。

図2.1に2進数の足し算、引き算、掛け算、割り算の例をひとつずつあげておきましょう。

$ \begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \\ \hline \end{array} $ <p>桁上げ 桁上げ (答え)</p> <p>足し算</p>	$ \begin{array}{r} 1\ 0\ 0\ 1 \\ -\ 1\ 0\ 1 \\ \hline 1\ 0\ 0 \\ \hline \end{array} $ <p>桁借り (答え)</p> <p>引き算</p>
$ \begin{array}{r} \\ \times 1\ 1\ 1 \\ \hline 1\ 1\ 1 \\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 1 \\ \hline \end{array} $ <p>桁上げ (答え)</p> <p>掛け算</p>	$ \begin{array}{r} \\ 1\ 0\ 0\ 0 \overline{) 1\ 1\ 1\ 1} \\ \underline{1\ 0\ 0\ 0} \\ 1\ 0\ 1 \\ \underline{1\ 0\ 0\ 0} \\ 1 \\ \hline 1 \text{ (余り)} \end{array} $ <p>(答え)</p> <p>割り算</p>

図2.1

実際のマイクロコンピュータでは上記の加、減、乗、除の算術演算（四則演算）はすべて加算で実現しています。すなわち減算は補数（後で説明します）の加算、乗算、除算は加減算の繰り返しで行なっています。

この他にマイクロコンピュータの演算で必要なものに論理演算があります。

その代表的なものをあげると次のようなものがあります。

- (1) 論理和 (OR) (\vee) : いずれか一方でも1のとき結果が1となる。

$$1 \vee 1 = 1, 1 \vee 0 = 1, 0 \vee 1 = 1, 0 \vee 0 = 0$$

- (2) 論理積 (AND) (\cdot) : 両方が1のときのみ結果が1となる。

$$1 \cdot 1 = 1, 1 \cdot 0 = 0, 0 \cdot 1 = 0, 0 \cdot 0 = 0$$

- (3) 排他的論理和 (XOR) (\oplus) : 片方が0で他方が1のときのみ結果が1となる。

$$1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 1 = 1, 0 \oplus 0 = 0$$

つぎに、2進数を10進数に変える方法を説明します。

10進数で187という数字を分解してみると次のようになります。

$$187 = 100 + 80 + 7 = 1 \times 10^2 + 8 \times 10^1 + 7 \times 10^0$$

これと全く同様な考え方をすれば2進数は次のように10進数に変換できます。

たとえば 1011_2 は

$$\begin{aligned} 1011_2 &= 1000_2 + 000_2 + 10_2 + 1_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 8 + 0 + 2 + 1 = 11 \end{aligned}$$

となります。

1001_2 は

$$1001_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 0 + 0 + 1 = 9$$

です。このようにしてどのような2進数も10進数に変換することができます。

逆に10進数を2進数に変える方法について説明します。たとえば10進数の27を2進数に変えてみます。その方法は27を次々に2で割っていくのです。図2.2を見てください。

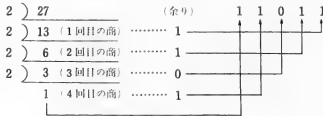


図2.2

このようにして計算して、商が1になったら、そこから矢印の方向に1, 0を並べます。この結果は 11011_2 となります。これが2進数で表された10進数の27です。

ここでちょっとためしに先程覚えた2進数を10進数に変換する方法でこの 11011_2 を10進数にしてみます。

$$11011_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 0 + 2 + 1 = 27$$

答えは27になりました。この例に従って自分で適当に問題を作って2進数↔10進数の変換をマスターして下さい。

2進数の桁の単位としてビット (Bit ; Binary = 2進の Digit = 桁, の略号) という言葉を使います。たとえば 101_2 は3ビットの2進数, 1101_2 は4ビットの2進数という使い方をします。コンピュータにおいては、2進数をいくつかまとめて処理します。そのことを一度に4ビットの処理をするコンピュータとか8ビットを処理するコンピュータであると表現します。

コンピュータにはこのように一度に処理できるビット数としては、4ビット、8ビット、16ビット、32ビットなどいろいろありますが、我々がここで扱う TK-85 では一度に8ビットの処理をします。つまり、マイクロコンピュータにもいろいろあり TK-85 というマイクロコンピュータでは8ビットを1単位として処理するものであるということです。なお8ビットのことを1バ

2章 マイクロコンピュータの中をのぞく

イト (Byte) ともいいます。そのため一度に8ビットの処理をするコンピュータのことをバイトマシンということもあります。

さて、2進数もビットが少ないうち、つまり桁数が少ないうちはよいのですが桁数が増えてくると大変見にくくなってきます。10111101₂などと書いてあっても読みにくいし、書くのも大変です。コンピュータにとっては有利なことの多いこの2進数も人間にとっては扱いにくくなってきます。

そこで、10111101₂などを4ビットずつに区切って、それぞれの4ビットに対して16個の数字を対応させる16進数というものがよく使われます(4ビットで16通りの文字が表現できます)。つまり10111101₂を

1011 1101₂

のように分け、表2.1の変換表に従って16進数にするのです。すなわち1011₂はB、1101₂はDと表せます。したがって10111101₂は16進数で2桁のBDという数になりますね。

表 2.1

2 進 数	10 進 数	16 進 数
0 0 0 0 0 0 0 0	0	0
0 0 0 0 0 0 0 1	1	1
0 0 0 0 0 0 1 0	2	2
0 0 0 0 0 0 1 1	3	3
0 0 0 0 0 1 0 0	4	4
0 0 0 0 0 1 0 1	5	5
0 0 0 0 0 1 1 0	6	6
0 0 0 0 0 1 1 1	7	7
0 0 0 0 1 0 0 0	8	8
0 0 0 0 1 0 0 1	9	9
0 0 0 0 1 0 1 0	10	A
0 0 0 0 1 0 1 1	11	B
0 0 0 0 1 1 0 0	12	C
0 0 0 0 1 1 0 1	13	D
0 0 0 0 1 1 1 0	14	E
0 0 0 0 1 1 1 1	15	F
0 0 0 1 0 0 0 0	16	10
⋮	⋮	⋮
0 0 0 1 1 1 1 1	31	1F
⋮	⋮	⋮

この16進数は、マイクロコンピュータを使っていくうえには不可欠なものですから、早く慣れるようにして下さい。

(16進数では0～9の数字のほかにA B C D E Fといった文字が出ています。これは我々の使

っている数字は0～9までで、1桁で15まで表す数字がないため10以上の数字をアルファベットで代用したものです。したがってこのA B C D E Fは数字です。)

また、以後16進数ができたら、1AHのように末尾にHをつけて2進数や10進数と区別します。

次にマイクロコンピュータの扱える数の表現範囲の話をしします。TK-85では8ビット、つまり1バイトを1単位として処理することを書きました。

つまり一度に表現できる数の範囲は 00000000_2 から 11111111_2 まで、10進数でいうと0から255までの数なのです。ただし、これはマイクロコンピュータでは0から255までの10進数に相当するもののしかあつかえないわけではなく、プログラムをうまく作ることにより、どんな数字でも扱うことができますから心配はいりません。さて、10進数にすると0～255までの数を表現できると書きましたが、負数はどのようにして表現されるのでしょうか？ 幸いなことに数には正か負の状態しかありませんから、8ビットの2進数の最上位の桁（これをMSD: Most Significant Digit、といいます。）が0のとき正、1のとき負として、残りの7ビットで数を表すことにします。

こうすることによって、+127から-128までを表現できるようになります。ただし、負数の場合、MSDを1にただけではだめで、残りの7ビットは補数で表現されている必要があります。

さて補数という考え方は初心者にとっては大変難解だと思います。

これは2進数を扱うコンピュータ（ディジタルコンピュータといいます）の宿命といってもよいでしょう。これはコンピュータの減算回路を加算回路で実現しようという考えによるためです。わかりやすくするためまず10進数で説明します。

10進数1桁の数aの10の補数とは $a + b = 10$ となるbのことです。すなわち、1と9、2と8、3と7、4と6、5と5が互いに補数の関係にあります。

この補数の関係を用いて減算を加算で行なってみましょう。8-4の場合を考えてみます。

8-4を行う代りに $8 + (4 \text{の補数})$ を考えます。ここで4の補数は6ですから $8 + 6 = 14$ となります。ここで桁上げの1を無視すれば8-4と同じ結果になります。すなわち減算の場合、引く数の補数を加えて桁上げを無視すればよいことになります。しかし桁上げが起らないこともあります。この場合は結果が負の数になるときです。たとえば $4 - 8 = -4$ は、補数を用いて計算すると $4 + 2 = 6$ となります。そしてこの6の補数4が求める数の絶対値です。

それでは2進数による補数の表し方について説明しましょう。

10進数の-8をディジタルコンピュータで表現するとき、まず8を2進数にします。

$8 = 00001000_2$ (1バイト表現)

そしてすべてのビットを0なら1へ、1なら0へ変えます。

$00001000_2 \Rightarrow 11110111_2$

そしてその結果に1を加えます。

$$\begin{array}{r} 11110111_2 \\ + \quad \quad 1_2 \\ \hline 11111000_2 \end{array}$$

この11111000₂が補数の考え方による-8なのです。つまり、負数表現をしたい10進数の絶対値の数を2進数にして、すべてのビットを反転して1を加えることが2進数で補数表現をする手続きだと覚えておいて下さい。

では、この方法を使って先ほどの8-4=4および4-8=-4を2進数に直して計算してみましょう。

8-4の場合

8の2進数00001000₂を求めます。

4の2進数00000100₂を求め、さらにその補数11111100₂を求めます。

$$\begin{array}{r} 00001000_2 \\ + 11111100_2 \\ \hline 10000100_2 \end{array} \rightarrow 00000100_2 \text{すなわち10進数の4}$$

↑
桁上げを無視する

4-8の場合

4の2進数00000100₂を求めます。

8の2進数00001000₂を求め、さらにその補数11111000₂を求めます。

$$\begin{array}{r} 00000100_2 \\ + 11111000_2 \\ \hline 11111100_2 \end{array} \rightarrow 00000100_2 \rightarrow 10 \text{進数 } 4 \text{ の負の数 } -4$$

↑
補数を求める

↑
桁上げがないので負の数となります。

このようにして減算を加算として実行することができます。

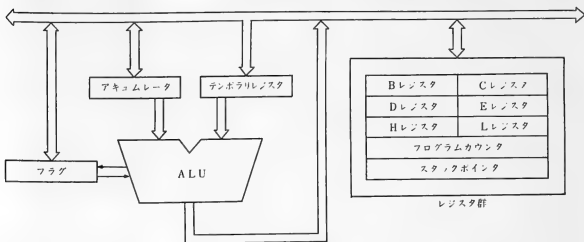
3. 8085 CPU の話

TK-85ではCPUに型番号が8085と呼ばれるCPUを使用しています。この8085というCPUは1974年アメリカのインテル社から発売された8080というCPUの改良発展型です。8080CPUは比較的早い時期に作られたCPUのため多くのユーザーに使われました。そのため、世の中にも広く使われているCPUとなっています。8085はこの8080をより進歩されたものですが、それはハードウェア（コンピュータ本体の回路）の改良であり、ソフトウェア（コンピュータの使用方法やプログラム）の点ではほとんど同じです。

そのため8080で作られたソフトウェアがほとんど変更なしに8085にも使用できます。人件費の高い現在では大変手間のかかるソフトウェアの開発にはばう大なお金がかかりますから、いまま

でに開発されたソフトウェアがそのまま自由に使えるということはユーザーにとっても大きな利点となります。

それでは、これから8080の発展型である8085の中味をのぞいてみましょう。



レジスタの構成

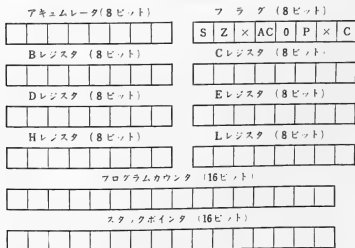


図2.3

8085 CPU は図 2. 3 に示したようになっています。以下主な部分を順番に説明することになります。

レジスタ——マイクロコンピュータがいろいろな仕事をするとき、メモリのほかにデータを一時的に記憶をする部分と考えて下さい。

8085では8ビットのB、C、D、E、H、Lレジスタをもっています。なお、BとC、DとE、HとLをペアにして16ビットのレジスタとして使用することもできます。

アキュムレータ——Aレジスタとも呼ばれ、各種の算術や論理演算等を実際に行うときに用い

られるレジスタです。B, C, D, E, H, Lレジスタとは多少異なった特徴をもっているためアキュムレータという名前で呼ばれます。

ALU——CPUの中で算術演算と論理演算を行うユニット(ALU: Arithmetic and Logical Unitの略)で、その入力の方にアキュムレータ、他方にテンポラリレジスタと呼ばれる作業用レジスタがあります。

プログラムカウンタ——マイクロコンピュータはプログラムに従って動いているわけですから、現在何番地のメモリに書いてあるプログラムを実行しているのかをマイクロコンピュータ自身が知らなければなりません。つまり次に実行すべきプログラムがメモリの何番地に書いてあるのかを知るもの、それがプログラムカウンタです。プログラムカウンタは16ビットのレジスタです。

スタックポインタ——これも16ビットのレジスタです。これはのちほど説明されるサブルーチンなどの利用法において、各レジスタの内容を保存しておくメモリの番地を示しておくものです。スタックポインタの詳しい使い方は後で述べます。

フラグ——これは8ビットのレジスタです。フラグレジスタはある特定の命令を実行したとき、その結果の状態によって1か0になるレジスタです。たとえば演算命令をしたときにフラグレジスタのZの部分は演算結果が0のとき1になり、そうでなければ0というように動きます。

以上、8085 CPUの中味の概略を述べてきました。

これらのアキュムレータやレジスタはプログラムにより示される命令を実行してゆくに従って次々に変化していきます。しかしその変化は決してあやふやな変化ではなく、同じ命令に対していつも同じように変化します。ですから、プログラムの中に、少しでも間違った部分があるとその後はマイクロコンピュータは(我々から見ても)メチャクチャな動きをしてしまいます。したがってプログラムは常に正確でなければいけません。

話をもとにもどします。

とにかく、TK-85に使われている8085 CPUの中味は以上述べたようになっていているということをよく覚えておいて下さい。

次にメモリ空間ということについてお話ししましょう。先程プログラムカウンタは16ビットのレジスタであると書きました。これはプログラムカウンタは16桁の2進数を表すことができるといことなのです。つまり、0000000000000000₂から11111111111111₂までの2進数を表現できるということです。10進数にしますと0から65535まで表現できることになり、メモリの番地を10進数で0～65535番地まで扱うことができます。8085は一度に8ビットの2進数を処理できます。つまり一度に1バイトを処理できますから、最大では65536バイトのメモリを操れることになります。

ただし標準のTK-85はROM 2048バイトRAM 1024バイトのメモリ装備ですが、マイクロコンピュータの勉強するにはこれで十分です。マイクロコンピュータの知識を身につけていくうちに、8085 CPUの最大メモリ空間である65536バイトまでTK-85を拡張することも可能になってきます。

4. 8085 CPU の命令について

メモリにはプログラムが書かれている話はすでにしました。しかし、メモリのある一部を見たときに、それが必ずしも CPU への命令が書かれているとはかぎりません。それはデータ（情報）かもしれません。つまりコンピュータのメモリに記憶されている内容は CPU に対する命令だけではないのです。

たとえば、コンピュータそのものはアイウエオの五十音や、ABCなどの文字をもっているわけではないのに、実際のコンピュータではいかにも扱われているように見えます。TK-85のCPU 8085では、一度に処理できる1バイト単位であらかじめ決められた約束事に従って文字に対応する2進数を決めておき（たとえば文字Aは2進数の01000001₂というように）、この2進数をメモリに必要に応じて記憶させておきます。そしてデータとしてAに対応する2進数ができたとき、これはAという文字だと判断していくのです。

このようにメモリの中にあるのは命令だけではないということを理解したうえで、プログラムの中における命令はどのようなものがあるかを2、3の例とともにお目にかかることにします。なお、命令の解説では16進数を多用します。マイクロコンピュータではプログラムでも16進数は大変よく使われますから16進数に慣れるようにして下さい。

まず、アキュムレタに4CHというデータを入れる命令について考えましょう。図2.4を見ながら以下の説明を読んで下さい。

TK-85 マイクロコンピュータではこの命令のことを MVI A, 4CHと表現（ニーモニックコード）され、メモリの0001H番地3E（0011110）として入っています。つまり、8085 CPUはプログラムを実行している最中にプログラムカウンタが示した番地にこの命令があったら、そのメモリの次の番地にあるデータ4CHをアキュムレタに入れるということです。

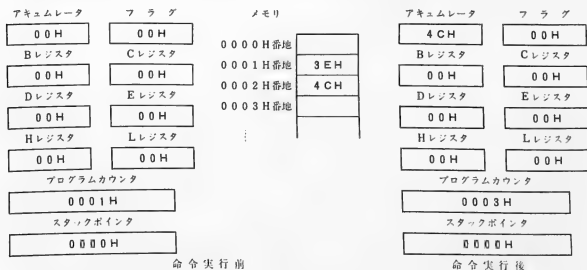


図2.4

おわかりでしょうか？ CPUはプログラムカウンタにより、次に実行する命令の入っているメモリ番地0001Hをおしえてもらい、そこに書いてある3EHを読み、その命令は、次のメモリ番地に書いてある内容をアキュムレータに入れることだと知り、それを実行します。そして、その実行が終わるとプログラムカウンタは次に実行する命令がはいっている0003Hを示しており、CPUは次の命令の実行のために0003Hの内容を読みにいこうとします。

言葉でいうと大変長くなりますが図と説明をよく読めばおわかりいただけると思います。

ここで一言ふれておきますが、この動作は人間の目から見ると一瞬のうちに終わります。このアキュムレータに4CHという内容を入れる命令を実行するのにかかる時間はわずか約 2.8×10^{-6} 秒、つまり1秒間にこの命令ばかりを繰り返すなら35万回もできるほど高速なのです。

もう一つの例を示します。

Bレジスタに入っているデータをDレジスタに移すことを考えてみます。この命令はメモリの中に16進数のかたちで50Hとして入っています。図2.5を見ながら以下を読んで下さい。

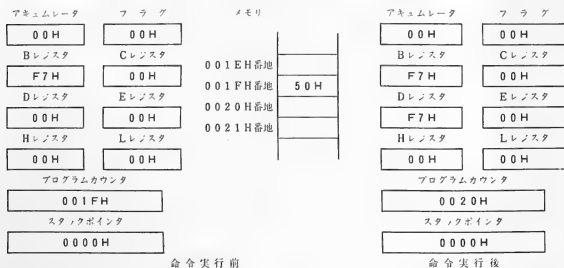


図2.5

この命令を実行する前にはBレジスタにはF7Hというデータが入っていて、Dレジスタの内容は00Hだったとします。プログラムカウンタが001FH番地を示したとき、CPUは001F番地に書いてある50Hという内容を読んで、その命令がBレジスタにはいっている内容をDレジスタに移すということを知りそれを実行します。その結果は当然ながらBレジスタの内容と同じデータがDレジスタに入ります。命令の実行後はプログラムカウンタはメモリの次の番地0020Hを指しています。おわかりですか？

ここで注意しておきますが、前のアキュムレータになんらかのデータを入れる命令とか、このBレジスタの内容をDレジスタに入れる命令などを実行しても、ほかのレジスタ類はなんの影響も与えないということです。この図2.4、図2.5の例では、命令の実行において直接関係のな

レジスタなどはすべて00Hにしていますが、レジスタの内容が00H以外でも同様なことがいえます。つまり命令に関係ないレジスタは変化しません。

さらにもうひとつの命令も見してみましょう。

アキュムレータにメモリの 1F3BH 番地に入っているデータを入れることを考えてみます。

図2.6を見てもらいます。プログラムカウンタは11FCH番地を指しています。CPUは11FCH番地の内容32Hを読み、その次の番地の内容を下位アドレス、そのまた次の番地の内容を上位アドレスとして、この上位、下位アドレスで示されるメモリ番地に書いてある内容をアキュムレータに入れる命令だということを知り、それを実行します。

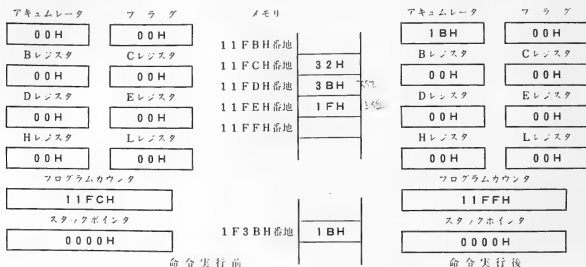


図2.6

以上の三つの命令についてながめてみましたが、こんなことに気付きませんでしたか？ 命令はいくつかの長さの種類があるのではないかということです。つまり、最初のアキュムレータになんらかのデータを入れる命令はメモリを2バイト使用しています。2番目のBレジスタの内容をDレジスタに移す命令はメモリを1バイト使っています。

このように命令にはメモリを1バイトだけ使う1バイト命令、2バイト使う2バイト命令、3バイト使う3バイト命令の3種があることをここでは覚えておいてください。

8085 CPU の命令はこのほかにもいろいろあります。しかし、他の命令もいま説明した三つの命令のようにそれほどむずかしいはありません。

読者の中には、「このような命令がいくつかあることはわかったけれども、どうしてこんな命令をすることで、コンピュータはあんなむずかしい仕事をすることができるのだろう」と考えておられる人もいるかもしれません。しかし、コンピュータは命令をうまく並べること、つまりプログラミングをすれば、単純な個々の命令からは想像もつかないむずかしい仕事もできます。その方法については次章以降で順次にお話ししていくことにしましょう。

3 章 TK-85 の操作方法

1. はじめに

1 章において TK-85 を自分の手で動かしてみました。しかし、その動かしかたは、ただ本書に書いてある操作手順に盲目的に従っていきただけで押したキーがどんな役割をもっているかを知っていたわけではありませんでした。この章では TK-85 のもっているそれぞれのキーの役割について説明していくことにします。

2. キーの説明をする前に

前にコンピュータはプログラムがなければ動かないことを説明しました。コンピュータはプログラムがなければなにもできないのです。では TK-85 は電源を ON にするだけでキー入力を受け付けてくれたり、LED にデータを表示してくれるのでしょうか？ 実は TK-85 は電源 ON と同時に動くプログラムをもっているからなのです。このプログラムにキー入力を受け付けてくれるプログラムや LED にデータを表示させるプログラムなどが入っているのです。0 章の終わりで説明した ROM にこのプログラムが入っています。ROM は電源を切ってもその内容をうしなうことはありません。ですからこのように電源が入ったら TK-85 がすぐに使えるようにするプログラムなどを入れるのに向いています。話をもどします。








このように電源 ON と同時に TK-85 をあるレベルまで操作しやすくしてくれるプログラム、このプログラムをモニタプログラムといいます。

そして、TK-85 がモニタプログラムを実行しているとき、TK-85 はモニタの管理下にあるといえます。

次に LED について述べます。以下の説明では LED の左側の 4 桁をアドレス LED、右側の 4 桁をデータ LED と呼びます（下図参照）



3. データ入力

TK-85 は電源を入れて  ボタンを押した直後、あるいは動作の途中に  ボタンを押した直後は必ずデータ入力可能な状態になっています。さて、データ入力とは     

5, 6, 7, 8, 9, A, B, C, D, E, F のそれぞれのキーを押して16進数のデータを TK-85 に入力することです。ではまずその操作をしてみます。

TK-85 の電源を入れ、**RESET** ボタンを押します。

以下を見て下さい。

キー入力	アドレス LED	データ LED
RESET	0000	0000
1/BC	0000	0001
2/DE	0000	0012
3/HL	0000	0123
4/SP	0000	1234
5/HP	0000	2345
6/HP	0000	3456

このように入力されたデータはデータ LED の右端から表示されていき、データ入力があるたびに左に桁上がりしていきます。そしてデータ LED がいっぱいになるとデータ LED の左端のデータはなくなりあとから入力した4桁のデータのみが有効になります。上図で見れば、**5/HP** のキーを入力したときには **2345** と表示されていますが、このときには **1/BC** の入力は無効となっていて、**2/DE** **3/HL** **4/SP** **5/HP** の入力だけが有効になっています。

4) NORMAL FUNCTION

TK-85 の操作は大きく分けて「NORMAL FUNCTION」、「MODE FUNCTION」、「REGISTER FUNCTION」の三つに分かれます。では最初に NORMAL FUNCTION から説明します。

4.1 ADDRESS SET AND MEMORY READ **ADDRESS SET**

このキーはデータ LED に表示されているデータをそのままアドレス LED に移し、そのデータを番地とするメモリの内容をデータ LED に表示します。

次の図を見て下さい。

...



この場合、データ LED には8123Hというデータが入力されています。そこで **ADDR SET** キーを押すと、そのデータはアドレス LED にそっくり移り、データLEDには8123H番地のメモリの内容である 3EH (これがなにになるかはあなたの TK-85 による、ここでは 3EH としただけです。)が表示されます。

4.2 ADDRESS INCREMENT AND MEMORY READ (**READ INC**)

アドレス LED に表示されているデータを+1して、その番地のメモリの内容をデータ LED に表示します。



上図ではアドレス LED に8123Hが表示されていたところへ **READ INC** を押したので、8123Hに+1した8124H番地がアドレス LED に表示され、データ LED にはその番地の内容 A6H (この内容も **ADDR SET** キーの説明と同じ)を表示します。

4.3 ADDRESS DECREMENT AND MEMORY READ (**READ DEC**)

このキーは4.2項の **READ INC** キーとは逆の働きをします。つまりアドレス LED に表示されているデータを-1して、その番地のメモリの内容を読み出すのです。その結果、アドレス LED には-1されたデータが、データ LED にはその番地の内容が表示されることになります。





上図ではアドレス LED には8123Hが表示されています。**READ DEC** キーを押すと8123Hから-1をしたデータをアドレス LED に、その番地のメモリの内容 7EH をそれぞれ表示します。

4.4 MEMORY WRITE AND ADDRESS INCREMENT (**WR ENT**)

アドレス LED の表示内容により指定されている番地のメモリに、データ LED に表示されている 下位2桁のデータを書き込みます。もし、指定されたメモリが書き込みのできない ROMで


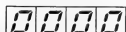
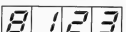




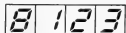
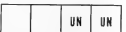

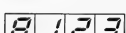
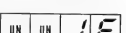


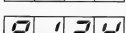
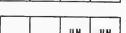

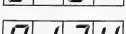
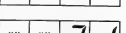


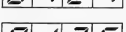
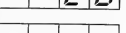

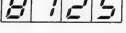
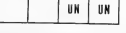
3章 TK-85 の操作方法



あったり、RAM が異常でメモリにうまくデータの書き込みができなかったりした場合は、アドレス LED に  と表示して、エラーがおこったことを示します。うまく書き込まれたときは続いて (このときは一瞬のうちに Rowe れます。) 先に説明した  キーの操作をしたのと同じことが Rowe れます。



キー入力	アドレス LED	データ LED
		
		

上図の場合、アドレス LED に表示されている 8123H 番地のメモリに、データ LED の下位 2 桁に表示されている 3EH という内容が書き込まれ、同時に 8123H に + 1 した 8124H 番地がアドレス LED に、そのメモリ番地の内容がデータ LED に表示されます。

- ここでいままて覚えたキー操作の応用をお目にかけます。メモリの 8123H 番地にデータ 1FH を、8124H 番地にデータ 2DH を書き込みたいとします。その操作手順はどのようにになるでしょう。操作手順は以下のとおりです。

キー入力	アドレス LED	データ LED
		
   		
		
 		
		
 		
		

本当にメモリに書き込まれたかどうかをたしかめるには  キーを使えばよいですね。上記のキー操作に続けて  キーを押せば、押されるごとにアドレス LED に表示されているメモリ番地がひとつずつ - 1 されていき、データ LED にその番地の内容が表示されますから、この場合ですと、8124H 番地に 2DH が、8123H 番地に 1FH が書き込まれているかがすぐわかります。

あるいは他の方法として、8123H 番地の内容を  キーを使って LED に表示させ、  キーを使うことも可能です。これは読者のみなさんが自分でやってみて下さい。

これまでの説明で、もうメモリにデータを書き込んだり、読み出したり、あるいは修正することができるようになりました。今まで説明してきませんでしたが、TK-85は標準装備でモニタプログラムにより自由に読み書きのできるメモリは8000H番地から8390H番地までです。読者のみなさんはこのメモリの範囲内でメモリにデータを書き込んだり読み出すことの練習をして下さい。

4.5 RUN (RUN)

この章の最初でモニタプログラムの話をしました。つまり我々が TK-85 のキーを押しているとき、TK-85 の 8085 CPU はこのモニタプログラムを実行しているわけです。2, 2, 4 項で書いたとおりに我々が TK-85 で自由にあつかえるメモリは8000H番地から8390H番地までです。

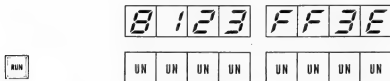
ではここに我々がプログラムを書いた場合、どのようにしてこのモニタプログラムから我々のプログラムへ 8085 CPU の実行を移すのでしょうか？、そのときのキーが RUN キーです。

アドレス LED に表示されているデータを番地とするメモリへ 8085 CPU の実行を移す、つまり、我々のプログラムを実行させるために我々のプログラムの書いてあるメモリ番地へ 8085 CPU を飛ばすのです。

キー入力

アドレス LED

データ LED



上図においては 8085 CPU は 8123H 番地から書いてあるプログラムを実行することになります。


4.6 RESET (RESET)

このキーを押すと TK-85 はどんな仕事をしていてもモニタプログラムの管理下にもどります。たとえば我々の書いたプログラムが少し間違っていたため、TK-85 がおかしい動作をしてしまったときなど、この RESET キーを押せばモニタの管理下にもどります。それからそのプログラムの間違った所を直したりすることができます。








TK-85 がおかしい動きをしたとき RESET を押すと TK-85 の頭が正常にもどると思っておいて下さい。

4.7 MONITOR (MON) と CONTINUE (CONT)

4.5 項で説明した RUN キーを使って我々のプログラムを TK-85 が実行しているとき、MON キーを押すと、8085 CPU のすべてのレジスタの内容を、ある所に保管しておきます。つまりアキュムレータ (Aレジスタともいう)、フラグ、B、C、D、E、H、L レジスタおよびプログラムカウンタ、スタックポインタの内容を保管しておくのです。そしてアドレス LED にはそのとき実行しようとしていたメモリ番地を表示し、データ LED には上位 2 桁にアキュムレータの 8 ビットのデータを16進数にした数字を、下位 2 桁にフラグの 8 ビットの内容を16進数にした内容を表示してモニタの管理下にもどります。

を押したときには逆に保管してあったレジスタ等のデータをすべて8085 CPUにもどし、先程実行していたプログラムを再開します。(ただし、プログラムの内容を変化させなかった場合は正常に動くが、それ以外の場合には保障できません。)

⑤. MODE FUNCTION ()

キーを押すとアドレス LED に四つの点が表示され、MODE FUNCTION に入ったことを知らせます。MODE FUNCTION に入ると以下六つのキー       のどれかを選択しなければなりません。

キー入力

アドレス LED

データ LED

8 1 2 3

F F 3 E






. . . .

5.1 ()

ここで説明します IN と次の 5.2 項の OUT は TK-85 のハードウェア (おもに回路などの意味) の知識が必要となりますので、まだ TK-85 のハードウェアについて詳しく知っていないかまたはこの本の11章を先に読んでからここを読んで下さい。ここではそれらの知識があるものとして書きます。

IN モードでは I/O ポート (INPUT & OUTPUT) からデータを入力するときに用います。I/O ポートとは入出力機器のことだと思われて結構です。

  と続けて押すと IN モードになり、アドレス LED にそれを表示します。

その後 IN PORT ADDRESS (つまり入力する機種の選択です。) を16進数2桁で入力し  キーを押すと IN PORT ADDRESS からデータを入力しデータ LED 下位2桁に表示します。

キー入力

アドレス LED

データ LED



. . . .



1 -

- -

1 -

F 8




1 - F 8

d F









1 - F 8

d F

上記では IN PORT ADDRESS に F1H を選びました。するとそこから入力されたデータ 74H がデータ LED 下位 2 桁に表示されました。続けて  を押すと新しい入力データの 76H が表示されました。(この入力データは別に意味はなく、みなさんの TK-85 ではどのように表示されるかわかりません。)

5.2 ()


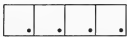


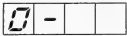



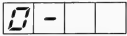
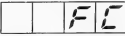

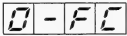



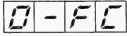
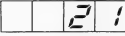

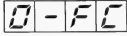
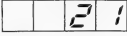
OUT モードは I/O ポートへデータを出力するときに使います。   キーを押すと OUT モードになりアドレス LED にそれを表示します。

続けて OUT PORT ADDRESS を 16 進数 2 桁で入力し、  を押し、さらに出力するデータを 16 進数 2 桁で入力して  を押すとそれが出力されます。ひき続き同じ OUT PORT ADDRESS にデータ出力をする場合には  を押すか、新しい OUT データを入力してから  を押します。

キー入力

アドレス LED





データ LED

上図では OUT PORT ADDRESS に FCH を選び、OUT データに 21H を選んで出力しています。

5.3 ()

このモードは、メモリに書いてある内容をそっくりそのまま違う番地のメモリに移す働きをするモードです。たとえば 8000H 番地から 8150H 番地までに書いてある内容を、8100H 番地から 8250H 番地までのメモリに移すなどです。

  と押すと MOVE モードになります。MOVE に対応するドット(点)だけが点灯してキー入力状態になります。まずメモリの中の移したい部分の先頭番地を 16 進数 4 桁で入力し  キーを押します。続いて最終番地を 16 進数 4 桁で入力し  キーを押します。このとき、先頭番地が最終番地より大きいときはエラーメッセージが出され、モニタプログラムの管理下にもどり、MOVE モードは解除されます。

3章 TK-85 の操作方法

次に移すメモリの先頭番地を16進数4桁で入力して **WR** を押すとメモリデータの転送が行われます。なお、この動作全体を **メモリのブロック転送** といいます。

さて転送が終了すると転送されたメモリの最終番地が表示され、MOVE モードのアドレス LED のドット表示が消えブロック転送が終わったことを示します。

キー入力	アドレス LED	データ LED
MODE	
MOV	. [] [] []	
ソース 先頭番地 → B AF AF AF	. [] [] []	8000
WR ENT	8.000	
ソース 最終番地 → B AF 5 AF	8.000	8050
WR ENT	. [] [] []	
宛先アドレス 先頭番地 → B 1 BC AF AF	. [] [] []	8100
WR ENT	8.100	
(転送終了)	8100	8150

7000H 転送
8000H
8050H
8100H
8150H

上図では8000H番地から8050H番地までのデータを8100H番地から8150H番地まで転送しています。

5.4 TEST MEMORY (**MODE** **TM**)

TEST MEMORY モードは TK-85 の RAM が正常に働いているかのチェックに使われます。将来、TK-85 の RAM を増設したときなどに使用されることもあります。

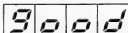
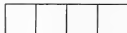
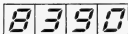
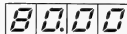
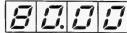
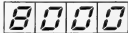
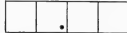
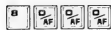
MODE **TM** と押すと TEST MEMORY モードに入ります。続いてテストするメモリの先頭番地を16進数4桁で入力して **WR** を押し、さらに最終アドレスを16進4桁で入力して **WR** を押すとテストメモリが始まります。テストが正常に終了するとデータ LED に good と表示、そうでないときにはアドレス LED にその異常のあったメモリ番地、データ LED の上位2桁に書き込んだデータ、下位2桁に読み出されたデータが表示されます。ひき続きテストメモリをするときには **WR** キーを押せば可能です。なお、このテストメモリを実行させるとその RAM の内容はデタラメになりますので注意して下さい。

アドレス LED
最終メモリ番地
データ LED
書き込んだデータ
読み出されたデータ

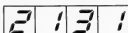
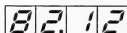
キー入力

アドレス LED

データ LED



テストメモリ終了



異常のとき

5.5 CASSETTE SAVE (MODE A/SAVE)

これはメモリの内容をカセットテープに録音するモードです。

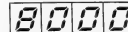
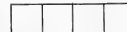
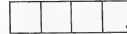
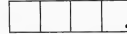
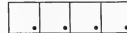
を押すと SAVE モードになります。

まず、録音するデータの名前にあたるファイル名を入力します。16進2桁で入力してを押します。これは00HからFFHまでの256通りがあります。次に SAVE するデータの先頭アドレスを16進4桁で入力してを押し、続いて最終アドレスを16進4桁で入力します。テープレコードとTK-85がしっかりつながっているかを確認したらテープレコードを録音状態にしてを押します。データ LED の最下位が点滅してSAVEが終了するとSAVEモードが解除されます。

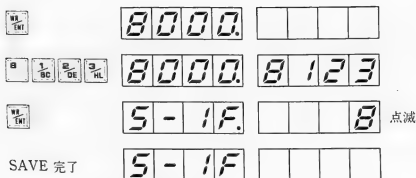
キー入力

アドレス LED

データ LED



3章 TK-85 の操作方法



上図では8000H番地から8123H番地までのデータを SAVE しています。








5. 6 CASSETTE LOAD (

と押すと LOAD モードとなります。

LOAD ファイルのナンバを16進2桁で入力します。 を押してテープレコードを再生させます。LOAD ファイルナンバーと異なるファイルナンバーを見つけるとデータ LED 下位2桁に表示します。LOAD したいファイルを見つけるとアドレス LED にその表示をしてロードを開始します。ロードが正常に終わるとロードしたデータの先頭番地と最終番地をそれぞれアドレス LED とデータ LED に表示します。ロードがうまくいかなかったときはアドレス LED にエラーメッセージを出します。

キー入力	アドレス LED	データ LED
(検索中)		(異なるファイルナンバー)
(ファイル発見)		(点減)
(終了)		
(異常あり)		

⑥. REGISTER FUNCTION (REG)

REGを押すとデータ LED の最上位のドットが点灯し、REGISTER FUNCTION に入ったことを知らせます。このモードになると次の七つのキー        の選択が必要になります。



キー入力

アドレス LED

データ LED



				.			

REG キー入力後、上記のいずれかのキーを押すとアドレス LED に表示レジスタ名、データ LED にその内容を表示します。表示レジスタの内容を変更したいときには16進データを入力して  キーを押します。16進データを入力せずに  キーを押すと次のレジスタ表示に移ります。

後で述べる BREAK 動作における BREAK POINT および BREAK DEPTH の設定はこの FUNCTION で行います。

キー入力

アドレス LED

データ LED



				.			
--	--	--	--	---	--	--	--



b	c	-	-	0	0	0	0
---	---	---	---	---	---	---	---



b	c	-	-	8	0	3	3
---	---	---	---	---	---	---	---



d	e	-	-	0	0	0	0
---	---	---	---	---	---	---	---

AT
BC
DE
HL
SP
BR-P
BR-D

上記においては TK-85 の 8085 CPU の BC レジスタの内容をデータ LED に 4 桁で表示します。B レジスタの内容は上位 2 桁に、下位 2 桁に C レジスタの内容を表示します。上記例では B レジスタに 80H というデータを、C レジスタには 33H というデータに修正したときの様子です。他のレジスタでも同様になります。読者のみなさんと試して下さい。なお、この FUNCTION で表示できるのは アキュムレータ & フラグ、B & C レジスタ、D & E レジスタ、H & L レジスタ、スタックポインタ、BREAK POINT、BREAK DEPTH の七つです。

⑦. ステップ動作

TK-85 のモニタプログラムでは、我々の作ったプログラムを 1 命令ずつ実行させることができます。これはプログラムが正しく書かれているかのチェック (デバッグ=虫と

ステップ動作

アドレス LED	データ LED	
PC の内容	A の内容	F の内容

3章 TK-85 の操作方法

り、ともいう)などに便利です。

これらの動作のことをステップ動作をさせるといいます。

ステップ動作を行うには、まず [RESET] キーあるいは [RUN] キーを押して BREAK DEPTH を 0 にする必要があります。 BREAK DEPTH については次の 8 節ブレイク動作で説明します。

なお、この BREAK DEPTH を 0 にするには 6 節で述べた REGISTER FUNCTION を使う方法もあります。BREAK DEPTH を 0 にしたらステップスイッチを STEP 側に倒します。この状態でアドレス LED にプログラムの先頭番地をセットして [RUN] キーを押しますとそのプログラムを 1 命令実行してモニタプログラムにもどります。

このとき、CPU のレジスタの内容はすべて保管されています。そしてアドレス LED にはプログラムカウンタの内容、データ LED の上位 2 桁にはアキュムレータの内容、下位 2 桁にはフラグレジスタの内容が表示されます。

この後 [CONT] キーを押しますと保管されていたレジスタの内容をすべて復元させ、次の 1 命令を実行して再びモニタプログラムにもどってきます。

さらに [CONT] キーを押すとこのステップ動作を引き続き実行できます。

6. ブレイク動作

BREAK POINT ... セットしたメモリ番地でプログラムの実行を中断する

BREAK DEPTH ... セットしたメモリ番地を何回通過したか中断するまでの回数

TK-85 のモニタプログラムでは BREAK POINT というものを設定することができます。

つまりこの BREAK POINT にセットしたメモリ番地においてプログラムの実行を BREAK する。つまり中断することができるようにになっているのです。さらに BREAK POINT のほかに BREAK DEPTH というものもっています。

これによって BREAK POINT にセットしたメモリ番地を何回通過したら BREAK するかもセットできるのです。

まず BREAK POINT と BREAK DEPTH を REGISTER FUNCTION を用いて設定します。この場合 BREAK DEPTH は必ず 1 以上にして下さい。

たとえば BREAK POINT ですぐにプログラムをストップしたい場合には、そのメモリ番地を BREAK POINT にセット、そして BREAK DEPTH を 1 にすればよいのです。

また BREAK POINT を何回か実行したあとで止めたいなら、その実行させたい回数に +1 した 16 進数を BREAK DEPTH に設定すればよいのです。なお、BREAK POINT にセットするメモリ番地は必ず各命令のオペレーションコードのあるメモリ番地でないといけなことに注意して下さい (このことについては次章を読んで下さい)。

以下に BREAK POINT と BREAK DEPTH の応用例を書いておきます。

図 3.1 ではオペレーションコードのことをオペコードと書いてあります。もしわからない場合は 4 章を読んでからここを読み直して下さい。

●BREAK動作の正しい使い方例

- (1) 指定BREAK POINTですぐ止めたい場合

```

[ BREAK POINT = 8008H ]
[ BREAK DEPTH = 1 ]

```

実行回数					
1	↓	8000	オペコード1	B ₂	
1		8002	"	2	B ₃ B ₂
1		8005	"	3	B ₃ B ₂
0		<u>8008</u>	"	4	
0		8009	J M P		8000H

- (2) 複数回実行後BREAK POINTで止めたい場合

```

[ BREAK POINT = 8008H ]
[ BREAK DEPTH = 3 ]

```

実行回数					
3	↓	8000	オペコード1	B ₂	
3		8002	"	2	B ₃ B ₂
3		8005	"	3	B ₃ B ₂
2		<u>8008</u>	"	4	
2		8009	J M P		8000H

●BREAK動作の悪い使い方例

- (1) プログラムの1命令ですぐに止める場合

```

[ BREAK POINT = 8000H ]
[ BREAK DEPTH = 1 ]

```

実行回数					
無限ループ	↓	<u>8000</u>	オペコード1	B ₂	
		8002	"	2	B ₃ B ₂
		8005	"	3	B ₃ B ₂
		8008	"	4	
		8009	J M P		8002H

- (2) オペコード以外が格納されているADDRESSを指定した場合

```

[ BREAK POINT = 8006H ]
[ BREAK DEPTH = 1 ]

```

実行回数					
無限ループ	↓	8000	オペコード1	B ₂	
		8002	"	2	B ₃ B ₂
		8005	"	3	<u>B₃</u> B ₂
		8008	"	4	
		8009	J M P		8000H

- (3) BREAK DEPTHが0の場合

```

[ BREAK POINT = 8008H ]
[ BREAK DEPTH = 0 ]

```

実行回数					
1	↓	8000	オペコード1	B ₂	
0		8002	"	2	B ₃ B ₂
0		8005	"	3	B ₃ B ₂
0		<u>8008</u>	"	4	
0		8009	J M P		8000H

図3.1

4章 プログラムの基礎と作り方

1. はじめに

1章で TK-85 を動かしてみました。TK-85 を動かしたプログラムは矢印プログラムというものでした。この章ではあの矢印プログラムはどのような手順で作られたか、どのような命令が使っているかを通じて 8085 CPU の命令体系や、プログラムの作り方などについて考えてみることにします。

2. プログラムとは何か

今までに何度か述べてきましたが、プログラムとはコンピュータの基礎的な命令をある法則に従って順序正しく並べてあるものです。そして、それだけではなく、コンピュータがこの命令の順序通りに実行していくと、与えられた仕事が確実にできるようになっていなければなりません。コンピュータという機械を働かせるには上記のプログラムというものの、つまり仕事の手順が書いてあることが必要です。

我々がプラモデルを作るときのことを考えてみましょう。プラモデルを作るにあたってまずその組み立て説明書を見ます。その説明書にはたとえば次のように書いてあるでしょう。

1. 部品に接着剤をつけない。
2. それを指定された所へとり付けなさい。
3. その上にシールを貼りなさい。

我々はこれを読んで、そのとおりにプラモデルを組み立てます。そのとき、組み立て説明書が間違っていたら、その間違い通りに組み立ててしまいます。

上の例を、我々がコンピュータ、組み立て説明書をプログラムに置き換えてみると非常に良く似ています。組み立て説明書が間違っていたら（プログラムが間違っていたら）とんでもないことになってしまいます。組み立て説明書がなかったら（つまりプログラムがなかったら）プラモデルの組み立ては不可能でしょう。

さて、コンピュータのできることは前にも書いたとおり基本的なもののばかりです。ですからコンピュータを使用してある仕事をさせるためのプログラムを作るとき、その仕事を複雑なものから簡単なものに、そして最後にはコンピュータのできる命令のレベルまで分解していきます。そこで仕事を分解できればコンピュータの命令をある法則に従って並べ、プログラムを作ることができます。

4章 プログラムの基礎と作り方

また、プログラムを作るときに、使用するコンピュータがどんな命令を持っているかを知らなくてはある仕事を分解していくときにも目安がありません。ですから、TK-85 でプログラムを作るときには、TK-85 の 8085 CPU の命令について知っておく必要があります。またその命令をいかに並べるかについても知っておかなければなりません。

1章において矢印プログラムの中味そのものをお見せいたしました。あの矢印プログラムも全体の仕事を細分化して TK-85 にできる命令まで分解して、それを並べたものなのです。

このようにプログラムを作るのは結構大変な作業です。しかし、このプログラムを作ることは今のところコンピュータにはできません。プログラムを作るには創造力や構成能力が必要なのです。コンピュータはそれらを持っていないか、あるいは非常に低いものしか持っていないのです。

ですから、今のところコンピュータのプログラムは人間が作る必要があります。ただし、そのプログラムを作る作業も人間に楽なようにするためのいろいろなアイデアがだされ、そして実用化されてきました。

たとえば、人間にとってわかりやすい文法に従ってプログラムを書くと、それをコンピュータにわかるようにしてくれるプログラムなどがあります。このプログラムのことを一般にはプログラム言語といっています。

このプログラム言語にはいろいろな種類があります。Fortran (フォートラン) や Cobol (コボル) などが有名です。マイクロコンピュータの世界では BASIC (ベーシック) というプログラム言語が盛んに使われています。TK-85 で使用できるプログラム言語はありません。直接、TK-85 にわかるようにプログラムを書きます。TK-85 に直接わかる言葉という意味で、このことをマシン語とか機械語といいます。つまり TK-85 では機械語を使用してプログラムを書きます。

機械語は人間にとっては不便なことの多い言語ですが、マイクロコンピュータを学ぶには最適な言語です。将来マイクロコンピュータを応用するときなどにも大変役立ちますから、早く慣れることが必要です。もちろん、TK-85 も拡張すれば機械語より便利なプログラム言語を使用することもできますから、あとでプログラム言語を使う楽しみを得ることもできます。

以上、プログラムについての概略をはなしてきましたが、いかがでしたか。プログラムについての概念が得られたでしょうか。

では次の節より 8085 CPU のもっている命令についての勉強をしていくことにします。

3. プログラムの基礎

3.1 フローチャート

プログラムの勉強をしていくとわかりますが、プログラムのアルゴリズム (考え方) を考えていく作業は簡単ではありません。それはプログラムのアルゴリズムを考えるときに必要とする要素が多いため、いきなりプログラムを作りはじめると、必要な部分が欠けてしまったり、不要な部分があたりしてしまいます。そこで、プログラムを作る前にそのプログラムで実現させる仕

事に必要な事項や順序などをはっきりさせ、図でもって整理しておく必要があります。そのためにフローチャート（流れ図）が使われます。

フローチャートとは簡単にいうと仕事のやる順序を記号や矢印などで表していくものです、図4.1を見て下さい。

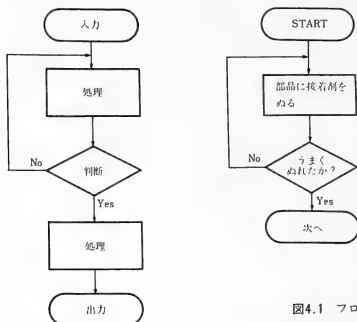


図4.1 フローチャート

左側が概念図、右側がプラモデルを作るときの様子をフローチャートにしてみました。

このような短い例ではフローチャートのありがたみはわかりにくいかもしれませんが、複雑な仕事をコンピュータにやらせるときなどフローチャートが書いてあるかどうかで、プログラム製作の能率は大きく異なります。

さらにフローチャートがあるとあとでプログラムを見直すときに大変便利です。同じようなことですが、他人の作ったプログラムを見るときにも便利です。フローチャートでプログラムの概略を見ながら、他人の作ったプログラムを解読するのもむずかしくありません。

このようにフローチャートはプログラムを作っていく上で非常に重要ですが、その使いかたは別にむずかしくありません。以下の文ではよくフローチャートがでできます。フローチャートというものに慣れるようにして下さい。

3.2 コーディングシート

プログラムを実際に作るとき、普通はまず紙と鉛筆を用いてそれにプログラムを書くことになります。その書き方にはいくつかのルールがあります。ルールといっても厳密なものではなく、いろいろな種類がありますが、原則的なものだけは覚えておかなくてはなりません。

プログラムを書く紙のことをコーディングシートといいます。図4.2を見て下さい。これがコーディングシートの見本なのです。いろいろな文字が書いてありますね。

4章 プログラムの基礎と作り方

FORTRAN PROGRAM SHEET					SHEET	OF
TK-85 矢印プログラム		Y. TAKEUCHI			DATE	1980. 1. 1
ADDRESS	LABEL	MNEMONIC	OPERAND	OP. CODE	REMARKS	
8000	START	LXI	H, 8358H	A15858		
8003		MVI	A, 8H	3E08		
8005		MVI	M, 40H	3640		
8007		CALL	8030H	C93080		
800A		MVI	M, 0H	3600		
800C		INX	H	23		
800B		DCR	A	3B		
800E		JNZ	8005H	C20580		
8011		JMP	8000H	C30080		
8030	SUB	MVI	D, 40H	1640		
8032		MVI	B, 0H	0600		
8034		DCR	R	05		
8035		JNZ	8039H	C23480		
8038		DCR	D	15		
8039		JNZ	8032H	C23280		
804C		RET		C7		

図4.2

図の左上から説明しましょう。

●ADDRESS (アドレス)

8085 CPU はプログラムをメモリに入れ、それを実行していきます。そのプログラムをメモリの何番地に入れるかを示しているのがこのアドレス欄です。8085 CPU のアドレス線は16ビットですので、16進数では4桁の数字で表すことができます。そのため、アドレスは16進数4桁で書かれます。

●LABEL (ラベル)

大きな仕事も細かくみると小さな仕事の積み重ねです。その小さな仕事の見出しを書くところです。プログラムが見やすくなりますし、プログラムを書くときにも大変役立つ欄です。

●MNEMONIC (ニーモニック)

8085 CPU の命令、つまり機械語は普通16進数で表示されます。しかし、人間には16進数の並びを見てそれがどのような命令かを知るのは大変むずかしいことです。そこで、人間にとってわかりやすい記号で命令を示すところ、それがニーモニック欄です。

●OPERAND (オペランド)

先のニーモニック欄には例外もありますが「何をする」の「する」にあたる部分しかありません。その動作の主体となる「何を」を表すのがこのオペランド欄です。

●OP CODE (オペコード)

ここには 8085 CPU の命令を16進数でそのまま書いてあります。TK-85 にプログラムを入力

するときにはこの欄を見ながら入力することになります。

ここに書いたコーディングシートの例は絶対ではありません。しかし、ここに書いたことを覚えておけばどんなコーディングシートを見ても理解できると思います。

4. 8085 CPU の命令体系

8085 CPU の命令解説

8085 CPU の命令は大きくわけて四つあります。

- (1) 転送命令
- (2) 演算命令
- (3) ジャンプ・サブルーチン命令
- (4) その他の命令

ではそれぞれの命令群の解説とその命令の代表例として矢印プログラムに使われている命令を解説することになります。

(1) 転送命令

転送命令とはレジスタとレジスタ間やメモリとレジスタ間での情報の移動についての命令のことです。

では、転送命令の代表について説明することにしたのですが、その前に知っておいてもらいたい記号がいくつかあるのでそれらを先に説明しておきます。

2章で少し説明しましたが8085 CPU 命令には1バイト命令、2バイト命令、3バイト命令の3種類があるといいました。つまり1バイトでひとつの命令を表す命令、2バイト、3バイトがひとかたまりになって命令を表す命令があるということです。この2バイト命令の2バイト目をこれ以後 B_2 と表し、同様に3バイト命令の2バイト目、3バイト目をそれぞれ B_2 B_3 と表すことにします。

それから r という文字を A (アキュムレータ)、B、C、D、E、H、L レジスタのうちの任意のものを表すとします。

また () はレジスタなどの内容を表すのに使います。たとえば (D) と書けば、D レジスタの内容を表すことになります。

それから $\langle B_2 \rangle$ 、 $\langle B_3 \rangle$ はそれぞれ命令の2バイト目の内容、3バイト目の内容を表します。そして [] は [] の中で指示されるメモリ番地の内容を示すことにします。

たとえば [(H)(L)] と書けば HL レジスタペアで示されるメモリ番地の内容、 $\langle B_2 \rangle \langle B_3 \rangle$ と書けば B_2 、 B_3 で指示されるメモリ番地の内容です。

ほかにもいろいろ説明しておきたい記号があるので、これくらいにして、転送命令の代表的な命令の解説をしていきましょう。

4章 プログラムの基礎と作り方

1. `MVI r, <B2>`

`(r) ← <B2>`

move immediate と呼びます。2 バイト命令で、<B₂> を任意のレジスタに入れることができます。

`MVI B, 3BH`

と書けば B レジスタに 3BH というデータを入力することになります。

2. `MVI M, <B2>`

`[(H)(L)] ← <B2>`

move to memory immediate の略です。これも 2 バイト命令で <B₂> を H, L レジスタで指示されるメモリ番地にしまいます。

たとえば H L レジスタペアに 83F8H というデータが入っているとき、

`MVI M, 30H`

と書くと、83F8H 番地に 30H というデータを格納します。

3. `LXI rp, <B3> <B2>`

`(rh) ← <B3> (rl) ← <B2>`

load register pair immediate の略です。rp とはレジスタペアのことです。rh とはレジスタペアの上位、rl とはレジスタペアの下位といいます。HL レジスタでいえば rh は H レジスタ、rl は L レジスタのことです。

3 バイト命令で <B₂> と <B₃> がレジスタペア、BC, DE, HL, SP レジスタのどれかに入ります。

`LXI H, 831FH`

と書くと HL レジスタペアに 831FH が入ります。

(2) 演算命令

演算命令を細分化すると次の三つになります。

1. インクリメント・デクリメント命令

2. 算術演算命令

3. 論理演算命令

1. のインクリメント・デクリメント命令はレジスタやメモリの内容を +1 したり -1 したりする命令です。この命令はキャリーを除くすべてのフラグを変化させます。ここでフラグの説明を少ししておきます。

フラグはある命令を実行したときにその結果の状態によって 1 か 0 になります。

たとえば演算命令を実行したときにその結果が 0 だと Z フラグが 1、そうでなければ 0。

キャリー（桁上げ：つまり答えが 8 ビットの最大値 1111111₂ を超えたときなど）がであれば CY フラグ = 1 となり、そうでなければ 0。最上位ビットが 1 なら S フラグ = 1、そうでなければ 0。

8ビットのレジスタの中味の1の数が偶数のときPフラグ=1, 奇数のとき0となります。

これらのフラグは我々がフラグの結果を必要とするとき以外にも次々に変化していますがそれらは気にしないで、フラグを見る必要のあるときだけフラグを見ればよいのです。

このフラグは後で説明する条件 JUMP のときなどに使われます。

2. 算術演算命令

これはA (アキュムレータ) レジスタを中心としてレジスタ間の和や差を計算します。

これによってフラグはすべて変化します。

3. 論理演算命令

A (アキュムレータ) レジスタを中心にレジスタ間の論理演算 (AND, OR, XORなど)を行います。フラグが変化します。

さて矢印プログラムで使用しているのは1. のインクリメント・デクリメント命令です。それは次の二つです。

1. INX rp
 $(rp) \leftarrow (rp) + 1$

increment register pair の略で、指定されたレジスタペアに+1します。フラグに影響を与えません。

2. DCR r
 $(r) \leftarrow (r) - 1$

decrement register の略で任意のレジスタから-1します。フラグが変化します。

(3) ジャンプ・サブルーチン命令

1. JMP
 $(PC) \leftarrow \langle B_3 \rangle \langle B_2 \rangle$

Jump 命令は3バイト命令です。〈B₃〉がプログラムカウンタの上位に、〈B₂〉が下位に入ります。この結果そのメモリ番地に CPU は実行を移します。

2. JNZ

Zフラグが0なら JMP 命令を実行します。Zフラグが1なら何もしないで次の命令に移ります。

3. CALL

Call 命令はそのときのプログラムカウンタの命令をスタックに入れて戻り番地として記憶し、スタックポインタを小さい方へ2動かし、アドレスのところで指定されている番地のところへジャンプします。サブルーチンへ飛ぶために使います。

$((SP) - 1) \leftarrow (PCH)$

$((SP) - 2) \leftarrow (PCL)$

$(SP) \leftarrow (SP) - 2$

$(PC) \leftarrow \langle B_3 \rangle \langle B_2 \rangle$

4章 プログラムの基礎と作り方

4. RET

(PCL) \leftarrow [(SP)]

(PCH) \leftarrow [(SP) + 1]

(SP) \leftarrow (SP) + 2

Return 命令でスタックの情報をプログラムカウンタに戻しサブルーチンからメインルーチンに戻ります。

(4) その他の命令

(1)~(3)の命令群のほかには割り込み命令やI/Oポートへの命令がありますがそれについては後で学ぶことになります。

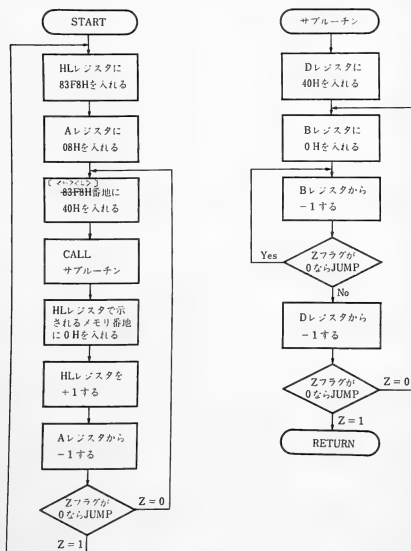


図4.3


以上大ざっぱですが 8085 CPU の命令体系を見てきました。次章以降でそれぞれの命令群の個々の命令について詳しく見ることにしましょう。

5. 矢印プログラムについて

4 節で覚えた命令を使えばもう 1 章で動かしてみた矢印プログラムを理解できます。この節では矢印プログラムを解説することにしましょう。

矢印プログラムのフローチャート

フローチャートをお見せする前に TK-85 の LED を操作するには TK-85 のメモリの 83F8H 番地から 83FFH 番地に適当な 16 進数 2 桁のデータを入力すればよいのです。

たとえば 83F8H 番地に 7FH というデータを入れると左端の LED が  になります。これだけを覚えておいて図 4.3 のフローチャートを見て下さい。

サブルーチンはウェイトループで時間つぶしのプログラムです。これを 8085 CPU のプログラムで書くと 1 章のリスト 1.1 になります。フローチャートをよく見ながらプログラムリストを見て、この矢印プログラムを解説して下さい。そして、1 章で少しプログラムを改造したところがありました。それがどのような意味を持っているかを考えて下さい。

6. アセンブラ言語について

アセンブラとはニーモニックコードで書かれたプログラムを自動的に機械的に翻訳するプログラムのことです。

TK-85 ではこのアセンブラ言語をもっていません。ですからニーモニックコードで書かれたプログラムを「インストラクション活用表」などによって手作業で機械語、つまり 16 進数に変換することになります。

この作業はプログラムを作ること自体には全く関係なく、ただの機械的な作業ですのでこれらはコンピュータにまかせることができるのです。

この考え方をさらに拡張すれば、プログラムの中で $\sin 30^\circ$ と書いただけで、それを機械語のレベルまで翻訳してくれるプログラムを作ること也有可能だということがわかります。このような言語は高級言語と呼ばれ、一般にはフォートラン、コボルなどの名前で見知られているものがそれにあたります。

この章では 8085 CPU を使ったプログラムの作り方についての解説をしました。8085 CPU のプログラムの基礎について、おわかりいただけたと思います。次章以降では、さらにプログラムの技術をみがいていくことにしましょう。

5章 データの取り扱い

1. はじめに

本章では、まずマイクロコンピュータを使うときの基本的なデータの取り扱い方として、レジスタ間、レジスタとメモリ間のデータの転送方法について、さらにデータをメモリに記憶したり、逆にそれを引用するのに、メモリの番地を直接に指定する方法と、間接に指定する方法について述べ、最後に、データの算術演算として加算・減算を、論理演算として否定・論理積・論理和・桁移動について述べます。

2. データの転送

2.1 レジスタ間のデータの転送

まず、レジスタ間におけるデータの取り扱いとして、1バイトの定数をAレジスタに入れる場合は、たとえば MVI (move immediate) 命令を用いて

```
MVI    A, 3
```

のように書きます。このようにすると定数3がAレジスタに入ります。ここでレジスタとしてはA、B、C、D、E、H、Lのいずれを用いてもかまいません。

次に特定のレジスタを他のレジスタに転送するには、たとえば、MOV (move) 命令を用いて

```
MOV     B, A
```

とします。このようにしますとAレジスタの内容がBレジスタに転送されます。このときAレジスタの内容は変化しません。

例5-1 定数23をBレジスタに入れ、それをCレジスタに転送するプログラムは右のようになります。

```
MVI    B, 23
MOV     C, B
HLT
```

このようにプログラムの最後には必ず一時停止命令 HLT を書いておきます。

上に述べた二つの命令は1バイトのデータしか取り扱うことができません。すなわち10進数の0から255までしか扱えません。それ以上のデータは2バイトを用いて表現する必要があります。2バイトのデータは二つのレジスタを対にして取り扱いますが、対にできるレジスタはきめられていて、BとC、DとE、およびHとLレジスタが対にして用いることができます。たとえば10進数の定数300を HL レジスタ対に入れる場合には LXI (load register pair immediate) 命令を用いて

LXI H, 300

と書き、また16進数の定数 ABCD を HL レジスタ対に入れるには

LXI H, 0ABCDH

と書きます。このときHレジスタに16進数の上2桁のAB、Lレジスタには下2桁のCDが入ります。これらの処理条件に書いたHはHLレジスタ対を表しており、HをB、D、SPと書きかえると、2バイトの定数がそれぞれBC、DEレジスタ対またはスタックポイントに入ります。

二つのレジスタ対にある2バイトのデータを交換するためにXCHG (exchange HL with DE) 命令

XCHG

があります。この命令を実行するとHLレジスタ対の内容とDEレジスタ対の内容が入れかわります。

2.2 番地の直接指定によるデータの記憶と引用

次にレジスタにあるデータをメモリに記憶する場合を考えてみましょう。それには記憶するメモリの番地を直接に指定する方法と、それを間接に指定する方法とがあります。ここでは前者について述べ、後者は2.3項で述べることにします。

まずAレジスタに保持している1バイトの定数を8100H番地のメモリに転送して記憶するためには、STA (store accumulator direct) 命令を用いて

STA 8100H

と書きます。

例5-2 定数5を8100H番地のメモリに
記憶するプログラムは右のようになります。

```
MVI A, 5
STA 8100H
HLT
```

逆に、メモリに記憶してある定数をレジスタに転送するには、たとえば、LDA (load accumulator direct) 命令を用いて

LDA 8100H

と書きます。このようにすると8100H番地のメモリにある定数がAレジスタに転送されます。

例5-3 8100H番地のメモリに定数5を
記憶してから、それを8200H番地に転送する
プログラムは右のようになります。

```
MVI A, 5
STA 8100H
LDA 8100H
STA 8200H
HLT
```

次に2バイトの定数を連続する二つの1バイトのメモリに記憶するためには、まずその定数をHLレジスタ対に入れてからSHLD (store HL direct) 命令を用いて

\$HLD 8100H

と書きます。このとき図5.1のようにLレジスタにある定数の部分が8100H番地のメモリに入り、Hレジスタにある部分が8101H番地のメモリに記憶されます。

例5-4	定数 300 を 8100H 番地と 8101H 番地を対にしたメモリに記憶するプログラムは右のようになります。	LXI	H, 300
		SHLD	8100H
		HLT	

3バイト以上のデータの場合は、2バイトずつに区切り、上の手順を繰り返し用いることができます。

例5-5	16進数の定数123456ABを8100H番地から始まる連続する4バイトのメモリに記憶するプログラムは右のようになります。	LXI	H, 1234H
		SHLD	8100H
		LXI	H, 56ABH
		SHLD	8102H
		HLT	

このプログラムを実行するとHLレジスタ対はまず「12 34」となり、次に「56 AB」となります。この結果メモリには図5.2のように記憶されます。

逆に、連続する2バイトのメモリに記憶してある定数をHLレジスタ対に転送するには、たとえばLHLD (load HL direct) 命令を用いて

LHLD 8100H

とします。このとき8100H番地と8101H番地を対にしたメモリに記憶してある2バイトの定数が、HLレジスタ対に転送されます。

例5-6	8100H番地、8101H番地のメモリを対にして2バイトの定数12ABHを記憶してから、それを8200H、8201H番地を対にしたメモリに転送して記憶するプログラムは右のようになります。	LXI	H, 12ABH
		SHLD	8100H
		LHLD	8100H
		SHLD	8200H
		HLT	

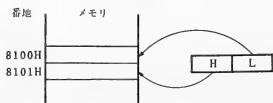


図5.1 命令 SHLD を用いて HL レジスタ対にあるデータをメモリに転送する

番地	メモリ
8100H	34
8101H	12
8102H	AB
8103H	56

図5.2 例5-5を実行したときのメモリの様子

2.3 番地の間接指定によるデータの記憶

データをメモリに記憶するのに、まずそのメモリの番地を一度 HLレジスタ対に入れてから、それを引用して記憶することもできます。たとえば、HLレジスタ対で指定したメモリの番地(M)に定数3を記憶するには、

```
MVI    M, 3
```

と書きます。

例5-7 間接指定の手法を用いて定数5を8100H番地に記憶するプログラムは右のようになります。

```
LXI    H, 8100H
MVI    M, 5
HLT
```

このプログラムは例5-2と同じ働きをします。

例5-8 8100H, 8101H番地のメモリに81A0H番地を記憶してから、その81A0H番地のメモリに定数ABHを記憶するプログラムは右のようになります。

```
LXI    81A0H
SHLD   8100H
LHLD   8100H
MVI    M, 0ABH
HLT
```

レジスタに入っているデータをHLレジスタ対に保持している番地のメモリに記憶するときには

```
MOV    M, A
```

と書きます。このときAレジスタに入っているデータがHLレジスタ対に保持している番地のメモリに記憶されます。この処理条件のAは他のレジスタ名でもかまいません。

例5-9 定数5をBレジスタに入れてから8100H番地に記憶するプログラムは右のようになります。

```
MVI    B, 5
LXI    H, 8100H
MOV    M, B
HLT
```

間接指定の手法を用いてAレジスタのデータをメモリに記憶したいときに、メモリの番地を指定するHLレジスタ対が使えない場合は、たとえばBCレジスタ対に記憶するメモリの番地をあらかじめ入れてから、STAX (store accumulator indirect) 命令を用いて

```
STAX   B
```

のように書くことができます。このようにしますと、AレジスタにあるデータがBCレジスタ対にある番地のメモリに記憶されます。処理条件のBをDに変えめすと、DCレジスタ対でメモリの番地を指定することができます。

例 5-10	定数 5 を A レジスタに入れてから、8100H 番地に記憶させるプログラムを、HL レジスタ対を用いなくて書きますと右のようになります。	<pre> MVI A, 5 LXI B, 8100H STAX B HLT </pre>
--------	--	---

このプログラムの第 2 行目と第 3 行目の処理条件の B を D に変えてもかまいません。

2.4 番地の間接指定によるデータの引用

メモリに記憶してあるデータを引用するときに、LDA 命令を用いてその番地にあるデータを直接引用しないで、いったんその番地を HL レジスタ対に入れてから、それを用いて A レジスタにデータを転送するには

```
MOV    A, M
```

と書きます。このとき HL レジスタ対に入れた番地のメモリの内容が A レジスタに転送されます。処理条件の A は他のレジスタ名でもかまいません。

例 5-11	あらかじめ 8100H 番地に定数 3 を記憶しておき、それを B レジスタに転送するプログラムは右のようになります。	<pre> MVI A, 3 STA 8100H LXI H, 8100H MOV B, M HLT </pre>
--------	---	---

HL レジスタ対が使えない場合、メモリに記憶してあるデータを間接指定の手法によって A レジスタに転送するには、たとえば BC レジスタ対に記憶するメモリの番地をあらかじめ入れてから LDAX (load accumulator indirect) 命令を用いて

```
LDAX   B
```

と書きます。このとき BC レジスタ対にある番地のメモリの内容が A レジスタに転送されます。この処理条件の B を D に変えることによりメモリの番地を指定するのに DC レジスタ対が使えます。

例 5-12	あらかじめ 8100H 番地に定数 3 を記憶しておき、それを A レジスタに転送するプログラムを HL レジスタ対を用いなくて書きますと右のようになります。	<pre> MVI A, 3 STA 8100H LXI B, 8100H LDAX B HLT </pre>
--------	---	--

このプログラムの第 3 行目と第 4 行目の B を D に変えてもかまいません。

最後にこの節で説明しましたデータの転送命令を分類すると、定数をレジスタまたはメモリに入れる場合、レジスタにある内容を他のレジスタに転送する場合、レジスタの内容をメモリへ、またはその逆にメモリの内容をレジスタへ転送する場合とがあります。これらの各場合の使用例

表 5.1 データの転送命令の使用例

分 類	使 用 例	動 作	注 意
定数 → レジスタ	MVI A, 3	3 → A	(注1)
	LXI H, 300	300 → HL	2 バイトの定数 (注2)
定数 → メモリ	MVI M, 3	3 → (HL)	
レジスタ → レジスタ	MOV B, A XCHG	A → B HL ↔ DE	(注3)
レジスタ → メモリ	STA 8100H	A → (8100H)	(注4)
	MOV M, A	A → (HL)	(注1)
	STAX B	A → (BC)	(注5)
	SHLD 8100H	H → (8101H), L → (8100H)	2 バイトの定数 (注4)
メモリ → レジスタ	LDA 8100H	(8100H) → A	(注4)
	MOV A, M	(HL) → A	(注1)
	LDAX B	(BC) → A	(注5)
	LHLD 8100H	(8101H) → H, (8100H) → L	2 バイトの定数 (注4)

(注1) AはB, C, D, E, H, Lでもよい。

(注4) 処理条件は2バイトの番地。

(注2) HはB, D, SPでもよい。

(注5) BはDでもよい。

(注3) B, AはA, B, C, D, E, H, Lのいずれでもよい。

を表5.1にまとめておきます。

3. データの算術演算

マイクロコンピュータで実行できるデータの演算としては、算術演算と論理演算の2種類があります。前者にはここで述べる加算と減算があり、後者には次節で述べる否定、論理積、論理和、桁移動などがあります。その他の演算は、これらの演算を組み合わせることで実行します。

3.1 加 算

2数の加算を実行する加算命令には以下で述べますように6種類ありますが、いずれの場合も加算結果はAレジスタに入ります。

まず、Aレジスタの内容に1を加えるときは INR (increment) 命令を用いて

INR A

と書きます。この命令の処理条件にはどのレジスタ名を書いてもよく、そのレジスタの内容に1が加わります。またメモリに記憶してあるデータに1を加える場合は、まずそのメモリの番地をHLレジスタ対に入れてから

INR M

と書きます。

レジスタ対の内容に 1 を加える場合は INX 命令を用います。たとえば INX (increment register pair) 命令を用いて、

INX H

と書きますと HL レジスタの内容に 1 が加算されます。この処理条件は B, D または SP と書いてもよく、そのときはそれぞれ BC, DE レジスタ対またはスタックポインタの内容に 1 が加算されます。

例 5-13	81A0H 番地から 81A2H 番地までのメモリに 10 から 12 までの定数を記憶するプログラムは右のようになります。	MVI	A, 10
		STA	81A0H
		INR	A
		STA	81A1H
		INR	A
		STA	81A2H
		HLT	

次に、A レジスタにある数値と 1 バイトの定数を加えて、それを A レジスタに入れるには ADI (add immediate) 命令を用います。たとえば、A レジスタの値に 3 を加えて A レジスタに入れるには

ADI 3

と書きます。また加える数がレジスタに前もって入っている場合には、ADD (add) 命令を用います。たとえば A レジスタと B レジスタにある二つの数値を加えて A レジスタに入れるときは、

ADD B

と書けばよいのです。この命令の処理条件としては A, B, C, D, E, H, L のいずれのレジスタ名を書いてもかまいません。なお、処理条件に A を書くと A レジスタの値を 2 倍することになります。さらに ADI, ADD 命令の代りに、ACI, ADC 命令を用いますと、上記の結果にさらに最上位のビットから桁上げがあったときに生じる CY フラグの値も加算されます。これは 2 バイトの数値の加算で桁上げがあるときなどに使用します。

例 5-14	3 + 2 を計算して 8100H 番地へ記憶するプログラムは右のようになります。	MVI	A, 3
		ADI	2
		STA	8100H
		HLT	

次に、A レジスタのデータとメモリに記憶してあるデータを加えるときには、あらかじめそのメモリの番地を HL レジスタ対に入れてから

ADD M

と書きます。さらに、ここに CY フラグの値も加えるときには、この ADD 命令の代りに ADC 命令を用います。

例5-15 2バイトの定数があり下位のバイトが8100H番地のメモリに、上位のバイトが8101H番地のメモリに記憶してあります。同様に8200H番地と8201H番地にも2バイトの定数が入っています。これらの定数を加えて、その結果を82ABH番地と82ACH番地に入れるプログラムは右のようになります。

```
LXI    B, 8100H
LXI    D, 82ABH
LXI    H, 8200H
LDAX   B
ADD     M
STAX   D
INX     B
INX     D
INX     H
LDAX   B
ADC     M
STAX   D
HLT
```

このプログラムでは2バイトの定数を上下のバイトに分けて、1バイト命令を用いて加算を行っています。すなわち、第6行目の STAX 命令までで下位8ビットの加算を行い、第7行目から上位8ビットの加算を行っています。このとき下位の加算で生じた桁上げを加えるために第11行目の ADC 命令を用いています。

最後に二つの2バイトのデータを直接加算できる DAD (double register add) 命令があります。まず一方のデータを HL レジスタ対に入れ、他方のデータを他のレジスタ対に入れてから DAD 命令を実行します。たとえば、HL レジスタ対と BC レジスタ対にある2バイトのデータを加算するときは DAD 命令を用いて

DAD B

と書きます。このとき結果は HL レジスタ対 に入ります。この命令の処理条件には B の他 D, H, SP と書いてもかまいません。

例5-16 2バイトの定数があり、下位のバイトが8100H番地のメモリに、上位のバイトが8101H番地のメモリに記憶してあります。それを2桁左へ移動するプログラムは右のようになります。

```
LHLD   8100H
DAD     H
DAD     H
SHLD   8100H
HLT
```

このプログラムは8100H番地のメモリに記憶してある値に、その値を加算し、さらにその結果を再度加算することによって2桁左へ移動しています。

<p>例 5-17 2 バイトの定数があり、下位の バイトが8100H番地のメモリに、上位のバ イトが8101H番地のメモリに記憶してありま す、この定数を5倍し、その結果を連続する 82EEH, 82EFH番地のメモリに記憶する プログラムは右のようになります。</p>	<pre> LHLD 8100H MOV C, L MOV B, H DAD H DAD H DAD B SHLD 82EEH HLT </pre>
--	---

このプログラムでは HL と BC レジスタ対に入れた定数を第4, 5 桁目の DAD 命令で4倍しています。第6行目の DAD 命令でその定数を4倍した結果に、再びその定数を加えていますので、結局5倍したことになります。

<p>例 5-18 DAD命令を用いて例5-15を 書き換えますと右のようになります。</p>	<pre> LHLD 8100H XCHG LHLD 8200H DAD D SHLD 82ABH HLT </pre>
---	---

今までに述べてきましたすべての加算命令をその使用例とともに表5.2に示しておきます。

表5.2 加算命令の使用例。ただし()内の命令を用いる
ときには、さらにCYフラグの値も加算される。

分 類	使 用 例	動 作	注 意
+ 1	INR A	$A + 1 \rightarrow A$	(注1)
	INR M	$(HL) + 1 \rightarrow (HL)$	
	INX H	$HL + 1 \rightarrow HL$	(注2)
加 算	ADI(ACI) 3	$A + 3 \rightarrow A$	(注1)
	ADD(ADC) B	$A + B \rightarrow A$	
	ADD(ADC) M	$A + (HL) \rightarrow A$	
2 バイトの加算	DAD B	$HC + BC \rightarrow HL$	(注2)

(注1) 処理条件はどのレジスタ名でもよい。

(注2) 処理条件はB, D, H, SPのいずれでもよい。

3.2 減 算

加算命令と同様に、減算命令も6種類あり、それらの減算結果はいずれもAレジスタに入ります。

Aレジスタの内容から1を引くときは DCR (decrement) 命令を用いて

DCR A

と書きます。この処理条件にはどのレジスタ名を書いてもかまいません。またメモリに記憶して

5章 データの取り扱い

あるデータから1を引く場合は、まずそのメモリの番地を HL レジスタ対に入れてから

DCR **M**

と書きます。

レジスタ対の内容から1を引く場合は DCX (decrement register pair) 命令を用います。たとえば、

DCX **H**

のように書きますと、HL レジスタ対の内容から1が引かれます。この処理条件はB、DまたはSPと書いてもかまいません。

減算命令にはAレジスタの内容から定数を引く SUI (subtract immediate) 命令、Aレジスタの内容からレジスタやメモリにある内容を引く SUB (subtract) 命令があります。これらは加算命令と同様にして使用できます。たとえば、Aレジスタの内容から定数3を引いてその結果をAレジスタに入れるには、

SUI **3**

と書き、またAレジスタの内容からBレジスタの内容を引き、その結果をAレジスタに入れるには

SUB **B**

と書きます。この処理条件には他のレジスタ名を書いてもよく、またAレジスタの内容からメモリの内容を引く場合は、あらかじめそのメモリの番地を HL レジスタ対に入れてから SUB 命令の処理条件をMと書きます。これらの使用例は表5.3にまとめておきます。

表5.3 減算命令の使用例。ただし()内の命令を用いるときには、さらにCYフラグの値も減算される。

分 類	使 用 例	動 作	注 意
- 1	DCR A	A - 1 → A	(注1)
	DCR M	(HL) - 1 → (HL)	
	DCX H	HL - 1 → HL	(注2)
減 算	SUI (SBI) 3	A - 3 → A	(注1)
	SUB (SBB) B	A - B → A	
	SUB (SBB) M	A - (HL) → A	

(注1) 処理条件はどのレジスタ名でもよい。

(注2) 処理条件はB、D、H、SPのいずれでもよい。

例5-19 82ABH番地のメモリに負の定数を記憶しておき、それを正になおして82ACH番地に記憶するプログラムは右のようになります。

注) 便宜上データを2進数で書く場合、小さな添字2をつけておきますが、プログラムのコーディング時にはこの書き方はできません。

```

MVI    A, 111110112
STA    82ABH
LXI    H, 82ABH
MOV     B, M
MVI    A, 0
SUB     B
STA    82ACH
HLT

```

3.3 2進化10進数による加算

いままで述べてきました命令の処理条件に書かれた定数は、プログラムを機械語になおして実行するときに、16進数でキーボードから入力する必要があります。また内部での処理は2進数で行われますが、その結果を外部に出力するときは1バイトを4ビットずつ区切って2桁の16進数の形で表示されます。そこでこの16進数を10進数に直す必要があります。この場合大きい数値の場合は、このような入力時と出力時における変換がめんどようになります。

加算の場合、この不便さをさける手法があります。まず16進数のキーボードを10進数のキーボードと見なしてそのまま入力しますと2進化と10進数の形で入力できます。二つの数をこのようにして入力したものを加算した結果をもう一度正しい2進化10進数に修正する必要があります。このために DAA (decimal adjust accumulator) 命令

DAA

用います。

たとえば $11 + 9 = 20$ の加算を行うとき、16進のキーボードで入力しますと 0BH, 09H と16進数になおさなくてははいけませんが、2進化10進数で加算する場合は、11と9をそのまま入力しますと、それぞれ 00010001_2 , 00001001_2 となって、2進化10進数で入力できます。しかしこの二つの数値を加算すると

$$00010001_2 + 00001001_2 = 00011010_2$$

となり、結果の下位4ビットが9を超えるので、2進化10進数になるように補正する必要があります。このとき DAA 命令を用いると、 00011010_2 が自動的に 00100000_2 に補正されます。

例5-20 11と9を2進化10進数で入力し、
その和を求めて8100H番地のメモリに記憶す
るプログラムは右のようになります。

```
MV I    A, 11H
MV I    B, 9
ADD     B
DAA
STA     8100H
HLT
```

3E 11
06 09
80
27
320081
46

この例では和が0から99までの10進数になる場合の加算しかできませんが、少し工夫すると大きな10進数の加算ができます。

例5-21 8100H番地と8101H番地のメモリに2進化10進数で記憶してある4桁の10進数の定数と、同じく8102H番地と8103H番地のメモリに記憶してある4桁の10進数の加算を行ない、その結果を2進化10進数で8104H番地と8105H番地のメモリに記憶するプログラムは右のようになります。ただし、8100H、8102H、8104H番地のメモリには下位の桁を記憶するものとします。

LXI	B, 8100H	010001
LXI	H, 8102H	210201
LXI	D, 8104H	110401
LDAX	B	0A
ADD	M	86
DAA		27
STAX	D	7C
INX	B	03
INX	H	23
INX	D	13
LDAX	B	0A
ADC	M	8E
DAA		27
STAX	D	12
HLT		76

このプログラムでは、まず第4行目と第5行目で10進数の下位2桁の加算を行っています。第6行目のDAA命令で和を正しい10進数に修正をしてから、第7行目のSTAX命令で8104H番地のメモリに記憶しています。同様に上位の2桁の加算を第12行目のADC命令で行っています。このときは下位の桁から桁上げも加算しています。このプログラムで、たとえば1234+5678を行ないますと、

番地	メモリ
8100H	34
8101H	12
8102H	78
8103H	56
8104H	12
8105H	69

のように記憶されます。

4. 論理演算

4.1 否定

Aレジスタにある1バイトのデータの各ビットの否定をとるときにはCMA (complement accumulator) 命令を用いて

CMA

と書きます。この命令を実行するとAレジスタにある8ビットの数値の1と0が入れ換わります。

例 5-22 Aレジスタに定数 5 を入れて、
その 2 の補数を求めて B レジスタに転送する
プログラムは右のようになります。

```

MVI    A, 5
CMA
INR    A
MOV    B, A
HLT

```

補数は各ビットの 1 と 0 を入れ換えて、その結果の最下位に 1 を加えれば求まりますので CMA 命令と INR 命令を用いて実行しています。その結果 B レジスタには 11111011₂ が入ります。

一方、CY フラグに 1 を入れるときは STC (set carry) 命令を用いて

STC

と書き、CY フラグの値の否定をとるときには CMC (complement carry) 命令を用いて

CMC

と書きます。このときレジスタや CY 以外のフラグの値は変わりません。

例 5-23 CY フラグの値を 0 にするプログラムは右のようになります。

```

STC
CMC
HLT

```

あらかじめ STC 命令で CY フラグの値を 1 にし、それを CMC 命令によって反転すると CY フラグに 0 が入ります。

4.2 論理積

値が 0 か 1 をもつ二つの 1 ビットの変数 x_1 , x_2 から論理積 x_1 and x_2 , 論理和 x_1 or x_2 , 排他的論理和 x_1 xor x_2 を求めると表 5.4 のようになります。1 バイトの二つのデータの対応する各ビットごとに、すべてのビットについて表 5.4 の論理積、論理和、排他的論理和の演算を行う命令があります。これらを用いるときは一方のデータは A レジスタに入れておく必要があります。また演算結果は A レジスタに入ります。これらの論理演算命令には A レジスタの内容と定数、他のレジスタまたはメモリの内容との間で演算を行うかによって、それぞれ 3 種類の命令があります。

表 5.4 2 変数の論理演算

x_1	x_2	x_1 and x_2	x_1 or x_2	x_1 xor x_2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

たとえば、A レジスタの内容と定数 3 H との各ビットの論理積を求めて、その結果を A レジスタに入れるためには ANI (and immediate) 命令を用いて

ANI 3H

とします。またAレジスタの内容とBレジスタの内容との間で論理積を求め、その結果をAレジスタに入れるためには ANA (and) 命令を用いて

ANA B

とします。この処理条件はA以外のどのレジスタを用いてもかまいません。また、あらかじめ HL レジスタ対でメモリの番地を指定しておき、その番地のメモリにある内容とAレジスタの内容の論理積を求めるときには命令 ANA の処理条件をMとすればよいのです。

例5-24	8100H番地にある2進数の定数	LDA	8100H
	が負の場合には、Aレジスタのビット7の値	ANI	10000000 ₂
	だけを1にするプログラムは右のようになります。	HLT	

4.3 論理和

論理和、排他的論理和の場合も同様で、Aレジスタの内容と定数との間で演算を行うためには、それぞれ、たとえば

ORI 3

XRI 3

のように、定数を処理条件に書いた ORI (or immediate), XRI (exclusive or immediate) 命令を用います。またAレジスタの内容とレジスタまたはメモリの内容との間で論理和の演算を行うためには、ORA (or) 命令を用いて

ORA B

と書きます。また排他的論理和の場合は XRA (exclusive or) 命令を

XRA B

表5.5 論理演算命令の使用例

分類	使用例	動作	変化するフラグ
論理積	ANI 3	A and 3 → A	Z, S, P CY, AC → 0
	ANA B	A and B → A	
	ANA M	A and M → A	
論理和	ORI 3	A or 3 → A	
	ORA B	A or B → A	
	ORA M	A or M → A	
排他的論理和	XRI 3	A xor 3 → A	
	XRA B	A xor B → A	
	XRA M	A xor M → A	

のように書いて用います。ただし、Mと書いてメモリの内容と演算するときにはあらかじめ HL レジスタ対にそのメモリの番地を入れておく必要があります。

最後に、いままで述べた論理積、論理和、排他的論理和命令とその使用例を表 5.5 にまとめておきます。ここで変化するフラグの項については次節で述べます。

例 5-25	8100H 番地にある 2 進数の定数	LDA	8100H
	が偶数なら、A レジスタのビット 0 を 1 にす	ORI	1
	ることによって奇数になおすプログラムは右	HLT	
	のようになります。		

例 5-26	2 進数の二つの定数 11001001 ₂	MVI	B, 11001001 ₂
	と 00001111 ₂ の論理積、論理和、排他的論理和	MVI	C, 00001111 ₂
	をとり、それぞれの結果を 8100H 番地から連	MOV	A, B ; AND
	続する三つのメモリに記憶するプログラムは	ANA	C
	右のようになります。	STA	8100H
		MOV	A, B ; OR
		ORA	C
		STA	8101H
		MOV	A, B ; XOR
		XRA	C
		STA	8102H
		HLT	

このプログラムでは、まず二つの定数を B、C レジスタに入れて、それぞれの論理演算をするときに、B レジスタから A レジスタにデータを移して用い、B レジスタのデータがこわれないようにしています。

表 5.5 の論理計算命令を繰り返し用いて二つ以上の定数に対する論理演算を行うこともできます。

例 5-27	2 進数の三つの定数 11001001 ₂	MVI	A, 11001001 ₂
	と 00001111 ₂ と 10101010 ₂ の論理積をとり、	MVI	B, 00001111 ₂
	結果を 8100H 番地に記憶するプログラムは右	MVI	C, 10101010 ₂
	のようになります。	ANA	B
		ANA	C
		STA	8100H
		HLT	

4.4 桁移動

Aレジスタの各ビットの値を順次左へ移動するときに用いられる命令として RLC (rotate accumulator left) 命令

RLC

と RAL (rotate left through carry) 命令

RAL

の二つがあります。図5.3のように RLC を実行すると、ビット7がビット0に直接移動します。このとき、ビット7の値が CY フラグにも入ります。この RLC 命令を何回用いて CY フラグの値が常に0ならばAレジスタの内容を2倍することができます。一方、RAL 命令の場合は CY フラグも含めて環状に連結されて、それらの各ビットの値が左へ移動します。

例5-28 ビット0から順次0が入るよう
にして1111111₂ を2回左へ移動し、その結
果を8100H番地のメモリに記憶するプログラ
ムは右のようになります。

```
MVI    A, 111111112
STC
CMC
RAL
CMC
RAL
STA    8100H
HLT
```



(a) RLC命令



(b) RAL命令

図5.3 左シフト命令によるデータの動き

また、このプログラムは ANI 命令を用いると次のように書き直すこともできます。

```
MVI    A, 111111112
RLC
ANI    111111102
RLC
ANI    111111102
STA    8100H
HLT
```


例5-29 定数5を10倍して、その結果を8100H番地のメモリに記憶するプログラムは右のようになります。

```

MVI    A, 5
MOV     B, A
RLC
RLC
ADD     B
RLC
STA     8100H
HLT

```

Aレジスタの数値を10倍するためには、はじめに RLC 命令を2回用いて4倍し、もとの数値を加えると5倍したことになります。次にもう一度 RLC 命令を用いると10倍することができます。

このようにAレジスタの値を2ⁿ以外の定数倍するときには、RLC 命令と ADD 命令を組み合わせ実行することができます。

次に、Aレジスタの内容を右へ1ビット移動する命令として RRC(rotate accumulator right) 命令

RRC

と RAR (rotate right through carry) 命令

RAR

の二つがあります。これらの機能は図5.4に示したように、RRC 命令を実行すると、Aレジスタの値が1ビットだけ右へ移動します。このときビット0の値はビット7とCYフラグに入ります。一方、RAR 命令はAレジスタにCYフラグを連結させて、それらの各ビットの値が全体として右向きに1ビット移動します。CYフラグの値を0としながらこのRAR 命令をn回用いますと、Aレジスタの内容を $\frac{1}{2^n}$ 倍することができます。なお、定数が割りきれなくて余りがでる場合は、それを無視します。

例5-30 定数11111110₂を4で割った結果を8100H番地のメモリに記憶するプログラムは右のようになります。

```

MVI    A, 111111102
RRC
ANI    011111112
RRC
ANI    011111112
STA     8100H
HLT

```

2回右へ移動し、ビット7、6に数値0を入れています。

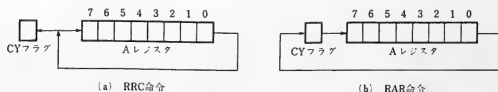


図5.4 右シフト命令によるデータの移動

6章 制 御 文

1. は じ め に

本章では演算結果がどのような状態であるかを判断して、それによって次に実行するプログラムの流れを決定する制御文について述べます。まずデータの状態を判断して表示するフラグの機能について述べ、次にそれによってプログラムの実行の流れを変える方法について述べます。さらにこれらの結果を用いて、特定の手段を繰り返し実行する方法を述べます。

2. 演算結果の判断

演算命令を実行して、その結果によって次に実行する文を決定したいことがあります。この場合、演算結果がどのような状態になっているかを表示するために8085では

Z, S, CY, AC, P

の5種類のフラグがあります。これらのフラグは図6.1に示すように状態レジスタの特定のビットを用いて表しています。各フラグの値は、強制的に設定するかまたは演算命令を実行したときに、自動的に表6.1のように1または0に設定されます。

5章の表5.2の加算命令、表5.3の減算命令および表5.5の論理演算命令を実行したときには、フラグの値は自動的に設定されます。たとえば

MVI A, 1
ADI 2

S	Z	0	AC	0	P	1	CY
---	---	---	----	---	---	---	----

図6.1 状態レジスタの各ビットとフラグの対応

表6.1 フラグの値の設定

フラグ	1	0
Z	演算結果が0のとき	演算結果が0でないとき
S	最上位の桁が1のとき	最上位の桁が1でないとき
CY	最上位桁から桁上げがあったとき	最上位桁から桁上げがなかったとき
AC	4ビットから5ビットに桁上げがあったとき	4ビットから5ビットに桁上げがなかったとき
P	8ビットの中に1が偶数個あるとき	8ビットの中に1が奇数個あるとき

として、ADI 命令によって 1 + 2 を計算するとフラグ Z, S, CY, AC, P はそれぞれ 0, 0, 0, 0, 1 となります。

ある状態を調べるのに表 5.3 の減算命令を用いてフラグの値を設定しますと、A レジスタの内容が変化してしまいます。減算結果は必要ありませんが、減算したときのフラグの値のみを用いたときには SUI や SUB 命令の代りに、CPI (compare immediately) 命令や CMP (compare) 命令を用います。

たとえば

CPI 3

としますと、A レジスタの内容から 3 を引いたときの結果によって Z, S, P, CY, AC フラグの値を設定しますが、減算結果は A レジスタに入らないので A レジスタの値はそのままです。また、A レジスタの内容から他のレジスタやメモリの内容を減算したときの結果によってフラグのみを変化させるには、たとえば

CMP B

と書きます。この処理条件の B は他のレジスタ名や M と書いてもかまいません。ただし、M と書いてメモリの内容と比較するときには、あらかじめそのメモリの番地を HL レジスタ対に入れておく必要があります。

例 6-1 CPI 命令によって CY フラグの値を設定する手法を用いて例 5-30 を書き換えると右のようになります。

```
MVI A, 111111102
CPI 0
RAI
CPI 0
RAI
STA 8100H
HLT
```

このプログラムでは CPI 命令を用いて CY フラグの値を 0 にしてから、RAI 命令によって左へ桁移動を行っています。

次にフラグの値を調べる方法を考えてみましょう。そのためには、状態レジスタの内容をみます。状態レジスタは他のレジスタのようにその内容を単独でメモリに転送することはできません。そこで状態レジスタと A レジスタを

A レジスタ	状態レジスタ
--------	--------

のように対にして、PUSH 命令を用いて

PUSH PSW

と書きますと、この 2 バイトのデータがスタックと呼ばれる補助メモリに転送されます。通常スタックの番地を保持する SP は初期値として、特定の番地を保持しています。TK-85 の場合は初期値は 8391H です。TK-85 で PUSH 命令を実行すると SP の値から 2 を引いた 838FH 番地の

メモリに状態レジスタの内容が入り、1を引いた 8390H 番地のメモリにAレジスタの内容が入ります。この様子を図6.2に示します。このように、状態レジスタの内容は 838FH 番地に入りますので、その内容を調べるとすべてのフラグの値がわかります。なお転送前の SP の値を任意に指定したいときは、あらかじめその値を SP に与えておく必要があります。たとえば8100H番地に状態レジスタの内容を入れたいときには

```
LXI    SP, 8101H
PUSH   A
```

とすればよいのです。

PUSH命令の処理条件としてAの外にB, D, Hを用いることができます。このときはそれぞれBC, DE, HL レジスタ対の内容がスタックに入ります。これは後述するように、サブルーチンを用いるときにレジスタの内容を一時保護するときなどに用いられます。

例6-2 1+2の計算をした結果、現れたフラグの値を調べるプログラムは右のようになります。	MVI	A, 1	3E01
	ADI	2	C602
	LXI	SP, 8102H	3102B1
	PUSH	A	F5
	HLT		76

ここで、8101H番地の内容を調べればよいわけです。

PUSH 命令を用いるとフラグとAレジスタの内容をSPで指定したメモリへ転送することができますが、逆にこのメモリの内容をレジスタ対にもどす場合、たとえば

```
POP     A
```

と書きます。このとき図6.3に示したようにSPであらかじめ指定した番地のメモリの内容が状態

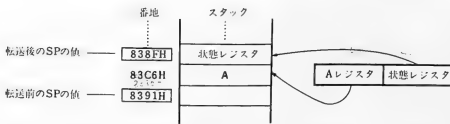


図6.2 PUSH 命令によって状態レジスタとAレジスタの内容をスタックへ転送する例

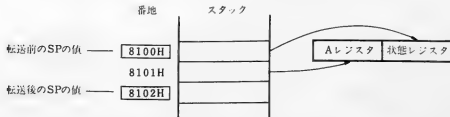


図6.3 POP 命令によってスタックからレジスタ対へ転送する例

レジスタに、その次の番地のメモリの内容がAレジスタに入ります。このとき SP の値は 2 が加算されます。したがって状態レジスタに適当な値を入れてフラグを任意に設定したいときは、あらかじめその値を SP で指定する次の番地のメモリに入れておきます。

POP 命令の処理条件として B, D, H を用いてもよく、このとき、スタックの内容はそれぞれ BC, DE, HL レジスタ対に転送されます。このように PUSH 命令と POP 命令を用いることによって、任意の 2 進数をスタックに一度入れてから、それを状態レジスタに転送することができます。

<p>例 6-3 すべてのフラグの値を 0 にする プログラムは右のようになります。</p>	<pre> MVI H, 00000010₂ PUSH H POP A HLT </pre>
--	---

まず第 1 行目の MVI で 2 進数 00000010₂ を H レジスタに入れ、それを第 2 行目の PUSH 命令によって現在の SP で指定されている位置から 1 を引いた番地のスタックに入れます。次に第 3 行目の POP 命令によって、それを状態レジスタに転送しています。

3. 実行流れの分岐

3.1 無条件分岐

プログラムは通常書かれた命令の順に実行されますが、途中である番地の文に分岐したい場合には無条件分岐命令 JMP (jump) を用いて

JMP 8100H

のように書きます。この文を書きますと 8100H 番地の文に実行が移ります。

<p>例 6-4 桁移動命令 RLC だけ を実行させない例として右のプログラムがあります。</p>	<pre> MVI A, 3 MVI B, 2 JMP JP RLC JP: ADD B HLT </pre>
--	--

3.2 条件付分岐

演算結果によって次に実行する文を指定したい場合は、先に述べた五つのフラグのうち四つのフラグを参照する表 6.2 に示した条件付分岐命令を用います。表 6.2 の I 群の分岐命令はフラグの値が 1 のときには、処理条件に書いた文に実行の流れは分岐し、0 のときには次の文を実行します。II 群の分岐命令は、フラグの値が 0 のときに処理条件に書いた文に分岐します。

たとえば、演算の結果、Z フラグが 0 になったとき、JNZ (jump non-zero) を用いて

表 6.2 フラグの値によって分岐する条件付分岐命令

分岐命令		参照するフラグ	フラグの値が1のとき	フラグの値が0のとき
I	JZ	Z	処理条件に書いたラベルの文へ	次の文へ
	JC	CY		
	JPE	P		
	JM	S		
II	JNZ	Z	次の文へ	処理条件に書いたラベルの文へ
	JNC	CY		
	JPO	P		
	JNP	S		

JNZ NEXT

⋮

NEXT: —

を実行しますと、JNZ 命令の次にラベルが NEXT の文に実行が移ります。

例 6-5 あらかじめ8100H番地に記憶してある定数が偶数ならそれを8101H番地に、奇数なら8102H番地に記憶するプログラムは右のようになります。 <i>偶数は奇数で判断する</i> <i>7-10-10</i>	LDA	8100H	<i>A ← (8100H)</i>
	MOV	B, A	<i>B ← A</i>
	ANI	1	<i>A ← A AND 1</i>
	JNZ	ODD	
	MOV	A, B	
	STA	8101H	
	HLT		
	ODD: MOV	A, B	
	STA	8102H	
	HLT		

このプログラムは第3行目の ANI 命令である定数の最下位桁と1との論理積をとり、定数が奇数なら結果は1になりますので、第4行目の JNZ 命令でラベルが ODD の文まで分岐します。

例 6-6 あらかじめ8100H番地に記憶してある定数の絶対値をとり、その結果を8101H番地に記憶するプログラムは右のようになります。 <i>絶対値を求める</i>	LDA	8100H	
	ADI	0	
	JP	PLUS	
	MOV	B, A	
	MVI	A, 0	
	SUB	B	
	PLUS: STA	8101H	
	HLT		

データの転送命令を実行してもフラグは変化しませんので、第2行目の ADI 命令でAレジスタの内容に0を加算してフラグを変化させてから、第3行目の JP 命令で符号ビットが0または正ならラベルが PLUS の文へ分岐し、符号ビットが1または負ならば、第4行目から第6行目で0

からBレジスタに入れた定数を引いて符号を反転しています。

3.3 3方向以上への分岐

演算結果によって2種類以上の分岐命令を用いますと、プログラムの流れを3方向以上に分岐させることもできます。

例6-7	8100H番地のメモリに	LD DA	8100H
	記憶してあるデータが正なら8101H	ADI	0
	番地、負なら8102H番地のメモリに	JZ	ZERO
	そのデータを記憶し、0なら8103H	JM	MINUS
	番地のメモリにFCHを記憶するプ	STA	8101H
	ログラムは右のようになります。	HLT	
	MINUS:	STA	8102H
		HLT	
	ZERO:	MVI	A, FCH
		STA	8103H
		HLT	

このプログラムの簡単なフローチャートを示すと図6.4のようになります。

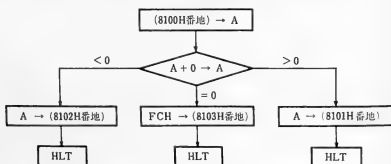


図6.4 データの値が正、0、負によって3方向に分岐する例

4. 繰り返し処理

前節で述べたプログラムの実行の流れを変える命令を用いますと、同じ一連の命令を繰り返し実行するプログラムを作ることができます。

例6-8	1から10までを加算し	MVI	A, 0
	て結果を8100H番地のメモリへ記憶	MVI	B, 0
	するプログラムは右のようになります。	MVI	C, 10
	LOOP:	INR	B
		ADD	B
		DCR	C
		JNZ	LOOP
		STA	8100H
		HLT	

このプログラムはカウンタとしてCレジスタを使っています。すなわちCレジスタに10を入れておき加算することに1を引いていきます。そして、Cレジスタの値が0になるまで第4行目から第6行目までの演算を繰り返し行っています。

例6-9	例6-8のプログラム	MVI	A, 0	3F 00
で1から10までの加算を行っている		MVI	B, 0	06 00
途中で和が30をこえれば、それ以上		MVI	C, 10	07 0A
加算を行わずに、その結果を8100H	LOOP:	CPI	30	FE 1E
番地のメモリに記憶し、もし30をこ		JP	OVER	7D 00
えなければ加算結果を8101H番地の		INR	B	04
メモリに記憶するプログラムは右の		ADD	B	80
ようになります。		DCR	C	05
		JNZ	LOOP	C2 08
		STA	8101H	32 00
		HLT		
	OVER:	STA	8100H	
		HLT		

このプログラムは第4行目のCPI命令でAレジスタに入っている和と30とを比較して、和が30をこえると第5行目のJP命令によって繰り返し演算からはずれてラベルがOVERの文まで分岐します。

例6-10	10×3を計算して、結	MVI	A, 10	3E 0A
果を8100H番地のメモリに記憶する		MVI	B, 2	06 02
プログラムは右のようになります。	LOOP:	ADI	10	C6 0A
		DCR	B	05
		JNZ	LOOP	C2 08
		STA	8100H	32 00
		HLT		

このプログラムは第3行目から第5行目までで10に10を2回加算しています。

例6-11	30÷3の商を計算して、	MVI	A, 30	
結果を8100H番地のメモリへ記憶す		MVI	B, 0	
るプログラムは右のようになります。	LOOP:	INR	B	
		SUI	3	
		JNZ	LOOP	
		MOV	A, B	
		STA	8100H	
		HLT		

このプログラムは第3行目から第6行目までで、結果が0になるまで30から3を順次減算し、減算した回数をBレジスタで数えています。

例 6-12 定数38を2進数で表したときに1となるビットの数を求め、それをAレジスタに入れるプログラムは右のようになります。

```

MVI    A, 38
MVI    B, 8
MVI    C, 0
LOOP:  RAL
        JNC    JUMP
        INR    C
JUMP:  DCR    B
        JNZ    LOOP
        MOV    A, C
        HLT

```

このプログラムは、RAL 命令を用いて1回左へ桁移動することに第5行目の JNC 命令で、CY フラグが1なら第6行目の INR 命令でCレジスタに1を加え、CY フラグが0ならラベルが JUMP の文まで分岐しています。このことを8回繰り返しますと定数38の2進数表示において1となるビットの数が求まります。

⑤ データの入出力

マイクロコンピュータの CPU にインターフェスを介して入出力装置を接続しますと、それを通じてデータの入出力を行うことができます。入出力用のインターフェイスは CPU との間にデータ、制御信号、番地を転送するための共通バスをもち、そのほかにいくつかの入出力装置を接続する端子があります。たとえば、8255と名付けられた入出力インターフェイス用 LSI を用いて入出力装置を接続した様子が図6.5に示されています。8255の場合は3台までの8ビットの入出力装置A、B、Cを接続することが可能です。なお、端子Cは上位の4ビットと下位の4ビットを独立に用いることもできます。以下では8255を用いる場合の入出力について述べます。

三つの入出力装置を用いる前に、まずどの装置をどのように用いるかを指定しておく必要があ

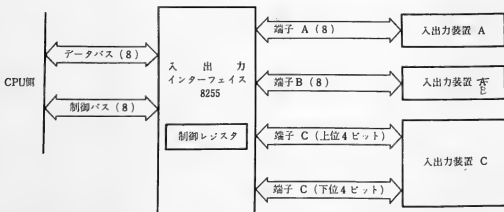


図6.5 入出力インターフェイス用 LSI 8255 と CPU との接続。()内はビット数を示す。

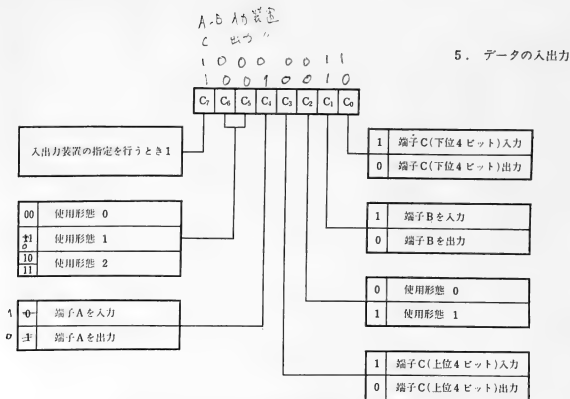


図6.6 8255 による入出力装置の指定信号と制御レジスタ

表 6.3 8255による入出力装置の使用形態

使用形態	意 味
0	三つの装置を入力または出力装置として使う。
1	装置 A, B を入力または出力装置として、装置 C を制御用に使う。
2	装置 A を入力と出力装置に使う。

ります。たとえば装置 A を入力装置として装置 B を出力装置として用いるというような指示を与えておく必要があります。入出力装置の指示はインターフェイス用 LSI 内にある 8 ビットの制御レジスタ CR を用い、この各ビットに図 6.6 に示した入出力装置の指定信号を与えることによって行います。ここで入出力装置の使用形態としては表 6.3 に示すように 3 通りあります。たとえば装置 A, B を入力装置とし、C を出力装置として用いる場合は使用形態は 0 であり、制御レジスタを 10010010₂ (92H)

とすればよいのです。

制御レジスタに指定信号を与えるには、プログラムの最初にまず、その信号を A レジスタに入れてから、それを CPU からインターフェイスに出力する命令 OUT を用います。このとき、制御レジスタの装置番号は 0FBH ですから、これを OUT 命令の処理条件に書きます。たとえば、装置 A, B を入力装置とし、C を出力装置として用いる場合は、プログラムの最初に

```
MVI    A, 92H
OUT    0FBH
```

と書いておきます。

以上のような準備ができましたら、次にある装置からデータを CPU に入力する場合には

IN 装置番号

と書きます。ここで装置番号は表6.4に示したように各装置に付けられた番号で、たとえば、装置 A から入力する場合は

IN 0F8H

と書きます。このとき、1バイトのデータが入力装置 A から A レジスタに転送されます。また出力命令としては

OUT 装置番号

があります。この命令によって A レジスタにあるデータが装置番号で指定された出力装置に転送されます。

表 6.4 8255の装置番号

装 置 名	A	B	C	CR (OUTのときのみ)
装置番号	0F8H	0F9H	0FAH	0FBH

一方、外部の装置に1ビットの制御信号を送るときには端子Cを用いることができます。そのときまず端子Cを出力装置として指定してから、端子Cのどのビットに1または0の信号を送るかを指定するために図6.7の指定コードの値を表6.4に示してある装置番号を用いて CR へ送ります。たとえば端子Cのビット7の値を1にしたいときには

MVI A, 0FH

OUT 0FBH

と書いて、制御レジスタに

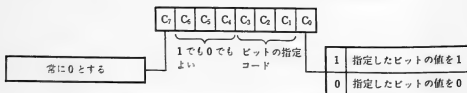


図6.7 端子Cの各ビットに1または0を指定する制御信号



図6.8 アナログ信号を A/D 変換してディジタルデータを CPU に転送する例

$$-\frac{1}{2}(\frac{1}{2})^2 \in \frac{1}{2}(\frac{1}{2})^2 \text{ d'après (1)}$$

7章 サブルーチンの考え方

1. はじめに

プログラムの流れの中で、まとまった機能を持つ部分があたり、何度も用いる所がある場合には、その部分をプログラムの流れからとりだして独立させておいて、これを引用する形でプログラムを書くと、全体のプログラムの流れが整理されます。このとき独立させた部分をサブルーチンといい、最初に実行する文のある主となるプログラムを主プログラム（メインプログラム）といいます。この章ではサブルーチンの定義とそれを引用する手法について述べることにします。

サブルーチンの効用はいろいろあります。たとえば、プログラムの中で四則演算のプログラムはコンピュータに計算をさせる場合に大変多く使われます。もし、四則演算をするたびに、この四則演算プログラムを繰り返し使用していたのではプログラムが大変長くなってきます。そこで四則演算プログラムをひとつの独立したプログラムとしておいて、必要に応じて呼び出す、つまりサブルーチンプログラムとしておけばプログラム全体の長さも短くなります。

ほかに、サブルーチンを利用するとプログラムを作り易くなるという利点があります。つまり、プログラムを作る場合に、おおまかにプログラムを考えておきます。それからプログラムの細かい部分はサブルーチンプログラムとして作っていくのです。このようにすると、プログラムが構造化されプログラムを作るのが大変楽になります。

このように有用なサブルーチンもそれを利用する場合にいくつか知っておかなければならないことがあります。次節からそのことについて述べていくことにします。

2. サブルーチンの定義

2.1 無条件リターンのサブルーチン

サブルーチンには主プログラムからそれを引用した場合、その実行が終われば無条件に主プログラムに戻る無条件リターンのサブルーチンと、サブルーチンで処理した結果によって主プログラムに戻る条件付きリターンのサブルーチンとがあります。無条件リターンのサブルーチンは、その先頭の文にラベル（標識）を付け、最後の文には、主プログラムに戻るための RET (return) 命令を書きます。

```
SUB: _____  
:  
RET
```

例7-1	Aレジスタにある定数	SUB:	RLC	
	を2回左へ桁移動して、そのあとは		ANI	FEH
	最下位桁から順に0がはいるプログ		RLC	
	ラムをサブルーチンにすると右のよ		ANI	FEH
	うになります。		RET	

このサブルーチンは、RLC 命令で1ビット左へ環状に桁移動して、ANI 命令で最下位桁を0にしています。そして最後の RET 命令で実行の流れは、無条件に主にプログラムに戻ります。

2.2 条件付きリターンのサブルーチン

前の例のようにサブルーチンの最後に RET 命令があれば、実行の流れは無条件に主プログラムに戻ります。ところが、サブルーチン内で処理した結果によって、主プログラムに戻るかどうかを決定したいことがあります。このときに使われるのが表7.1に示した条件付きリターン命令です。これらの命令はフラグの値によって主プログラムに戻るかどうか決定します。戻らない場合はその命令を無視して次の命令を実行します。

たとえばサブルーチン内の演算の結果、Zフラグが0になったときに主プログラムに戻るためにはサブルーチン内に RNZ 命令を含み

SUB: _____

⋮
RNZ
⋮
RET

の形に書きます。このようにすると、RNZ 命令を実行するときにZフラグが0なら主プログラムに戻りますが、1のときは RNZ 命令の次の命令を順次実行して最後の RET 命令によって主プログラムに戻ります。

例7-2	8100H番地のメモリに	SUB:	LDA	8100H
	記憶してある定数の絶対値をとって		ADI	0
	Aレジスタへ転送するサブルーチン		RP	
	は右のようになります。		MOV	B, A
			MVI	A, 0
			SUB	B
			RET	

Aレジスタに8100H番地のメモリの定数を転送して、それが正なら第3行目の条件付きリターン命令によって主プログラムに戻り、負なら第4行目から第6行目までを正になおしてから主プログラムに戻ってきます。

表 7.1 フラグの値によって主プログラムに戻る条件付きリターン命令

フラグ	Z	S	CY	P
0	RNZ (return if not zero)	RP (return if positive)	RNC (return if no carry)	RPO (return if parity odd)
1	RZ (return if zero)	RM (return if minus)	RC (return if carry)	RPE (return if parity even)

例 7-3 0 以外の二つの 1 バイトの定数の掛算を行うサブルーチン
MLTY は右のようになります。ただし二つの定数は A レジスタと B レジスタにあらかじめ入っているものとして、また答は 2 バイトの定数で HL レジスタ対に入れるものとします。

```

MLTY: MOV    E, B
      MVI    D, 0
      MOV    L, D
      MOV    H, D
      LOOP:  DAD    D
      DCR    A
      JNZ    LOOP
      RET

```

このサブルーチンは、a を b 回加算して $a \times b$ の掛算を行っています。A レジスタをカウンタとして利用して、最初そこに b を入れておき第 5 行目から第 7 行目までの文によってカウンタの値が 0 になるまで HL レジスタ対に a を加算しています。

例 7-4 正の定数 a, b (ただし $a \geq b$) の割算 $a \div b$ を行うサブルーチン DIV は右のようになります。ただし、a は A レジスタに、b は B レジスタに入っているものとします。また商は C レジスタに、余りは D レジスタに入れることにします。

```

DIV:  MVI    C, 0
      LOOP:  INR    C
      SUB    B
      JZ     JUMP
      CMP    B
      JP     LOOP
      JUMP:  MOV    D, A
      RET

```

このサブルーチンは、A レジスタの値から B レジスタの値を、A レジスタの値が 0 か B レジスタの値より小さくなるまで引いていきます。引いた回数を C レジスタに入れておき、それを商とし、A レジスタに残った数を余りとしています。この例のフローチャートは図 7.1 に示されています。

条件付きリターン文を用いるサブルーチンにおいて、たとえば

```

SUB:      :
A         RNZ
          :
          JMP A

```

のように繰り返し処理を行う場合は、サブルーチンの最後に RET 命令を含まないこともあります。

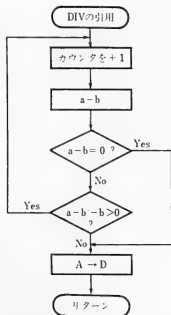


図7.1 例7-4のフローチャート

例7-5 1からCレジスタにある整数までを加算するサブルーチンは右のようになります。

```

SUB:  MVI    A, 0
      MVI    B, 0
LOOP: INR    B
      ADD    B
      DCR    C
      RZ
      JMP    LOOP

```

3. サブルーチンの引用

3.1 無条件引用

主プログラムにおいて、サブルーチンを引用するときには、CALL (call) 命令の処理条件にサブルーチンの先頭の文のラベル、たとえば SUB を用いて

```
CALL SUB
```

と書きます。なお、サブルーチンは主プログラムの前または後のどちらに書いてもかまいませんが、本書では主プログラムの後を書くことにします。

例7-6 主プログラムのA, Bレジスタに保持している定数2, 3をサブルーチンで加算してから左へ桁移動して, その結果を主プログラムによって8200H番地のメモリに記憶するプログラムは右のようになります。

```

MAIN: MVI    A, 2
      MVI    B, 3
      CALL   SUB
      LXI    H, 8200H
      MOV    M, A
      HLT

*
SUB:  ADD    B
      RLC
      RET

```

例7-7 例7-3を用いて 3×2 を計算し, 答えをAレジスタに入れるプログラムは右のようになります。

```

MAIN: MVI    A, 3
      MVI    B, 2
      CALL   MLTY
      MOV    A, L
      HLT

*
MLTY: MOV    E, B
      MVI    D, 0
      MOV    L, D
      MOV    H, D
LOOP: DAD    D
      DCR    A
      JNZ    LOOP
      RET

```

例7-8 8200H番地のメモリに記憶してある1バイトの2進数を, BCDに変換するプログラムは右のようになります。ただし, 例7-4の割算サブルーチンDIVを用います。

154

```

      STA    8200H
      CPI    100
      JM     JP1
      MVI    B, 100
      CALL   DIV
      MOV    H, C
      MOV    A, D
JP1:  CPI    10
      JM     JP2
      MVI    B, 10
      CALL   DIV
      MOV    A, C
      RLC
      RLC
      RLC
      RLC
      ADD    D
      MOV    L, A

```

	JMP	JP3
JP2:	MOV	L, A
JP3:	SHLD	8100H
	HLT	

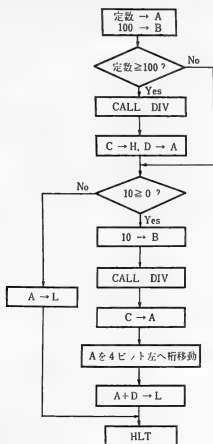


図7.2 例4-8のフローチャート

このプログラムではフローチャートに示した手順に従って2進数の 10000000_2 をBCDに変換しています。まず第6行目までで 10000000_2 を100で割り、商をHレジスタの下位4ビットに入れています。第16行目までで余りを10で割って、商をLレジスタの上位4ビットに入れています。第17行目のADD命令でその余りをLレジスタの下位4ビットに入れます。したがってBCDはHLレジスタ対に入り、SHLD命令でBCDは8100Hと8101H番地のメモリに記憶されます。

3.2 条件付き引用

主プログラム中の処理結果、すなわちフラグの値によってサブルーチンを引用したいことがあります。このときに用いられる条件付き引用命令が表7.2に示されています。たとえばZフラグが0のとき、ラベルがSUBのサブルーチンを引用するためには

CNZ SUB

と書きます。この命令は、Zフラグが1のときには無視されて、この次の文を実行します。この

3. サブルーチンの引用

ように CNZ, CNC, CPO, CP 命令はそれぞれに該当するフラグの値が 0 のときにのみサブルーチンを引用しますが、フラグの値が 1 のときは無視されます。同様に CZ, CC, CPE, CM 命令はフラグの値が 1 のときにのみサブルーチンを引用しますが、0 のときは無視されます。

例 7-9 Aレジスタに入れた定数が、偶数ならサブルーチンでその値と定数 3 とを加算して Aレジスタへ入れ、奇数ならそのままにしておくプログラムは右のようになります。	MVI	A, 定数
	MOV	B, A
	ANI	1
	CZ	SUB
	HLT	
	SUB :	MVI A, 3
		ADD B
		RET

表 7.2 フラグの値によって用いるサブルーチンの条件付き引用命令

フラグ フラグの値	Z	S	CY	P
0	CNZ (call if not zero)	CP (call if positive)	CNC (call if no carry)	CPO (call if parity odd)
1	CZ (call if zero)	CM (call if minus)	CC (call if carry)	CPE (call if parity even)

例 7-10 8100H 番地と 8101H 番地のメモリに記憶されている定数の和が、偶数なら 1 からその数までの和を求めて、奇数なら 1 を 8102H 番地のメモリに記憶するプログラムは右のようになります。	MAIN :	LDA	8100H
		MOV	B, A
		LDA	8101H
		ADD	B
		MOV	C, A
		ANI	1
		CZ	SUB
		STA	8102H
		HLT	
	*		
	SUB :	MVI	A, 0
		MVI	B, 0
	LOOP :	INR	B
		ADD	B
		DCR	C
		RZ	
		JMP	LOOP

このプログラムでは主プログラムの第 4 行目で加算を行い、その結果を第 6 行目と第 7 行目で調べて偶数なら SUB を引用し、奇数ならそのまま 1 を 8102H 番地のメモリに記憶するサブルーチンは例 7-5 の SUB と同じです。

4. 二つ以上のサブルーチンの引用

一つのプログラムの中にサブルーチンをいくつか書いてもよく、それらは必要に応じて何回でも使用することができます。またサブルーチンから他のサブルーチンを引用することもできます。

<p>例 7-11 Aレジスタにある定数が正または0ならサブルーチン1を引用して定数3を加算し、負ならそのままそれを主プログラムで8100H番地のメモリに記憶した後、再びサブルーチン2で左へ桁移動してその結果を8101H番地のメモリに記憶するプログラムは右のようになります。</p>	<pre> MAIN: MVI A, 定数 CPI 0 CP SUB1 STA 8100H CALL SUB2 STA 8101H HLT SUB1: MVI B, 3 ADD B RET SUB2: RLC RET </pre>
---	--

このプログラムの流れは図7.3に示したように条件によってサブルーチンを引用し、後でまたサブルーチン2を引用しています。

<p>例 7-12 例7-9で加算結果が6以上のときは、サブルーチン2で定数4を加算するプログラムは右のようになります。</p>	<pre> MAIN: MVI A, 定数 MOV B, A ANI 1 CZ SUB1 HLT SUB1: MVI A, 3 ADD B CPI 6 CP SUB2 RET SUB2: ADI 4 RET </pre>
--	--

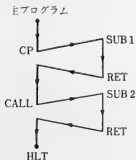


図7.3 サブルーチンを2回引用する例

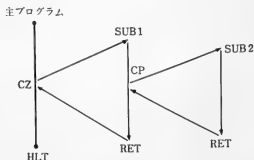


図7.4 サブルーチンから他のサブルーチンを引用する例

このプログラムは図7.4に示すように主プログラムから条件によってサブルーチン1を引用し、さらに条件によってそこから再びサブルーチン2を引用しています。

5. スタックの利用

5.1 サブルーチン実行後の戻り番地の記憶

サブルーチンを引用してその処理を終えれば、実行の流れはそれを引用したもとのプログラムの引用文の次に戻ります。そのため戻ったときに次に実行する文の番地を記憶しておく必要があります。この番地はサブルーチンを引用したときに、スタックと呼ばれるメモリ領域の2バイトを用いて自動的に記憶され、サブルーチン処理をすべて終えて、もとのプログラムに戻ったときに自動的にプログラムカウンタに転送されます。

スタックの番地はSP(スタックポインタ)と呼ばれる2バイトのレジスタを用いて指定します。SPはプログラムで指定されない限り、最初は特定の番地を示しています。たとえばTK-85の場合は初期値として8391H番地を保持しています。サブルーチンを引用する命令を実行すると、そのサブルーチンの処理を終了したときに戻るべき番地がSPにある番地のメモリの手前2バイトに記憶されます。このときSPの数値から2を引いた番地のメモリに下位1バイトが、1を引いた番地のメモリに上位の1バイトが入り、さらにSPの値は2が引かれて、次のスタックが利用できる状態になります。

またサブルーチンの引用が終わって実行がもとに戻ると、SPの値は2が加えられます。

例7-13 例7-6のプログラムを実行するとき、スタックとその番地を示すSPはどのようなになるでしょうか。

例7-6のプログラムは次のように8000H番地から始まるメモリに記憶してから実行するものとします。

番地				
8000H	MAIN:	MVI	A, 2	≧E 02
8002H		MVI	B, 3	06 03
8004H		CALL	SUB	CD 00B1
8007H		LXI	H, 8200H	21 00B2
800AH		MOV	M, A	77
800BH		HLT		76
8100H	SUB:	ADD	B	80
8101H		RLC		07
8102H		RET		C9

SPはあらかじめ指定しないかぎり、8391Hを保持しています。第3行目のCALL命令でサブルーチンSUBを引用するときには、その処理が終わって主プログラムに戻ったときに、次に実行される8007H番地が、スタックの838FH~8390H番地に図7.5のような順で記憶されます。

7章 サブルーチンの考え方



図7.5 サブルーチンを引用するときのスタックと SP の様子

このとき SP の値は2引かれて 838FH となります。サブルーチンの引用が終わると RET 命令により、スタックに記憶されている戻り番地がプログラムカウンタに転送され、SP の値は8391Hに戻ります。

スタックの内容を知るために用いる命令としては、XTHL (exchange H and L with top of stack) 命令

XTHL

があり、この命令は図7.6のようにSPで指定される番地のメモリの内容をLレジスタに、その次の番地のメモリの内容をHレジスタに転送します。このとき SP の値は変化しません。これと同様のことは POP 命令を用いて

POP H

と書いても実行できますが、このときは SP の値は2加えられます。

例7-14 例7-12を実行したときスタックにどのようなデータが入っていくかを調べる
プログラムは下のようになります。

番地			
8000H	MAIN:	MVI	A, 定数
8002H		MOV	B, A
8003H		ANI	1
8005H		CZ	SUB1
8008H		HLT	
8009H	SUB1:	XTHL	
800AH		SHLD	8100H
800DH		MVI	A, 3
800FH		ADD	B
8010H		CM ^o I	6
8012H		CP	SUB2
8015H		RET	
8016H	SUB2:	XTHL	
8017H		SHLD	8102H
801AH		ADI	4
801CH		RET	

このプログラムでは二つのサブルーチンを用いた場合に、スタックがどのように用いられるかを調べています。まず最初、SP は8391H 番地を示していますが主プログラムで SUB1 を引用したときには、戻り番地である8008H 番地が図 7. 7 (a)のように記憶されます。このことを確かめるために、8009H～800AH 番地の命令で、8008H を図 7. 7 (c)に示したように8100H～8101H 番地のメモリに記憶しておきます。

また SUB1 において、SUB2 を引用したときには戻り番地である8015H 番地は図 7. 7 (b)のように記憶されます。このときも同様に8016H～8017H 番地の命令で8015H を8102H～8103H 番地のメモリに記憶しておきます。主プログラムに戻るときには、逆に 838DH 番地から8391H 番地へと順次下へ戻ります。最後に8100H 番地から8103H 番地のメモリの内容を調べると SUB1, SUB2 を引用したときにスタックに入った値が分かります。

SP の初期値をプログラマが指定したい場合、たとえば8300H 番地としたい場合、6 章で述べましたように

LXI SP, 8300H

とすればよいのです。また HL レジスタ対に8300H 番地を入れておいてから SPHL (move H and L to stack pointer) 命令

SPHL

を書いて、HL レジスタ対の内容が SP に転送されます。

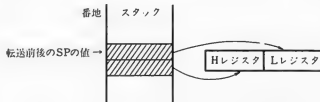


図7.6 XTHL 命令によるスタックの内容の読み出し

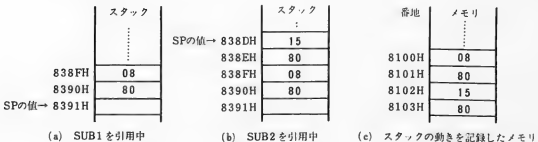


図7.7 例7-12の処理中の SP とスタックの動き

例 7-15	SP の値を8300H に指定してスタックを用いたいときは右のようにします。	LXI H, 8300H SPHL HLT
--------	--	-----------------------------

5.2 レジスタ対のデータの退避

8085 で用意されているレジスタ対は AF, BC, DE, HL の五つですが、もっと多くのレジスタ対を用いたいときや、またはサブルーチンを引用する前のレジスタ対のデータをサブルーチン実行中に一時保護しておきたいときにもスタックが利用できます。たとえば、HL レジスタの値をスタックへ退避するには

PUSH HL

と書くとよいのです。

→ 82FEH 30
82FFH 3A
8300H

例7-16 SPの値を8300H番地に設定してから、HLレジスタ対に入っている定数3A30Hをスタックに退避させるプログラムは右のようになります。

```
LXI SP, 8300H
LXI H, 3A30H
PUSH H
HLT
```

このプログラムでは 82FFH 番地に 3AH が、82FEH 番地に30Hが入っています。

例7-17 8100H～8101H番地に
ある定数10と5の加算を行いサブル
ーチンで減算を行った結果と加えて、
8103H番地に記憶するプログラムは
右のようになります。

```
LDA 8100H
MOV B, A
LDA 8101H
ADD B
PUSH PSW
CALL SUB
POP PSW
LXI H, 8102H
ADD M
STA 8103H
HLT
SUB: LDA 8101H
MOV B, A
LDA 8100H
SUB B
STA 8102H
RET
```

このプログラムは第6行目の PUSH 命令でAレジスタの値をスタックへ一時退避して、サブルーチンを引用した後で、またスタックの値をAレジスタに戻して用いています。

6. 割り込み処理

いままでに述べたサブルーチンは、あらかじめ、決められたときに引用してプログラムの流れを変更しましたが、あるプログラムを実行中に周辺機器から信号を加えることによってプログラムの流れを、別の処理を行うサブルーチンに変更することができます。この処理を割り込み処

理といい、これを行う命令を割り込み命令といいます。この処理は周辺機器とのデータの転送などに用いられます。

主プログラムにおいて割り込まれてもよいときは EI (enable interrupt) 命令

EI

を書き、割り込まれては困る場合は DI (disable interrupt) 命令

DI

を書いておきます。

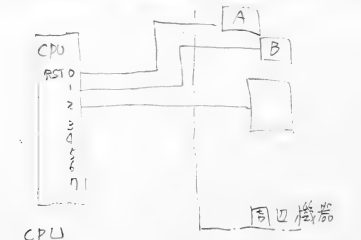
あるプログラムを実行中に割り込みが発生するまでの経過は次のようになります。まず割り込みたい周辺機器から CPU へ割り込み要求信号が送られてきます。そのとき EI 命令があって CPU が割り込み可能ならば、現在実行中の命令の処理が終わった直後に割り込みができる状態になります。次に CPU から周辺機器へ割り込み確認信号が送られ、データベースが割り込みを要求した機器に対して開放されます。このとき、サブルーチンのときと同様に、戻り番地は自動的にスタックに記憶されます。次に周辺機器から、割り込み実行命令である RST (restart) 命令

RST n

がデータベースに乗せられて CPU に入ってきます。この命令は各周辺機器にそれぞれ別の処理条件をもつようにハードウェアで用意する命令であり、プログラム中にはできません。処理条件の n は 0 から 7 までの 8 種類までの整数がとれます。

RST 命令は、無条件分岐命令と同じ機能を持ちます。すなわち、割り込みをする周辺機器からその機器に割り当てられた RST 命令がくると、プログラムの流れをいったん分岐します。

割り込みが発生しているときには他の割り込みは受け付けられず、複数の割り込みが同時に入ってきたときは、RST 命令の処理条件の小さい数を用いるほど優先順位が高くなります。



① 割り込み要求信号

② EI 命令が有り
割り込み可能

③ 割り込み確認信号

④ データベースが割り込み要求の機器に開放

⑤ RST n 命令 (ハードウェアで決められている)

⑥ 割り込み処理

8 章 モニタの詳細な説明

1. はじめに

どんなコンピュータもプログラムなしでは、何も仕事をすることができません。これらの処理プログラムはコンピュータのメモリ内部に書かれ、実行されます。したがって、これらの処理プログラムをメモリに対してアクセスするような基本的な機能やデバッグ機能等が必要になります。

モニタプログラムは、一般的に処理プログラムに対して制御プログラムと呼ばれ、各処理プログラムを効率良く管理し、コンピュータの中核機能となるプログラムです。大型計算機には OS と呼ばれる巨大な管理プログラムが存在しますが、モニタはこの OS のマイクロコンピュータ版だと思ってください。

TK-85 のモニタは 2 k バイト の大きさと構成されています。全体の構成を理解していただくために 2 節以降にフローチャートおよび各部分の詳細を説明します。3 章に述べた、操作方法を頭に浮かべながら一つ一つ理解して行って下さい。

2. フローチャート

モニタ全体の構成を把握していただくために図8.1にフローチャートを示します。

リスティングを解説する際に全体の構成が分かっていると、部分的な解説ができて、それぞれのつながりがうまくいかず、結局断片的な理解で終わってしまいます。したがっていきなりリスティングの解説を行わずに、まずこのフローチャートによって全体を把握してください。

〈フローチャートの用語説明〉

(AR) : アドレスレジスタ——2 バイト

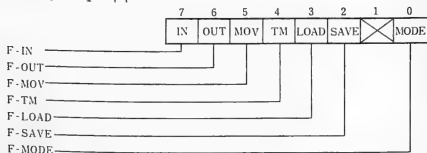
(DR) : データレジスタ——2 バイト

(PC) : プログラムカウンタセーブエリア——2 バイト

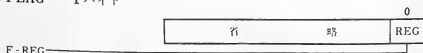
(BD) : ブレイクデプス——2 バイト

(BP) : ブレイクポイント——2 バイト

MODE FLAG——1 バイト



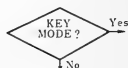
REG FLAG——1 バイト



○F-MOVのビットテスト
ON : "1"
OFF : "0"



○MODE FLAG 8 ビットすべてのビットテスト
ON : NON 0
OFF : 0



○キー入力データの判別

3. メモリマップ

3.1 システムメモリマップ

アドレス	容量(バイト)	ROM or RAM	備 考
FFFF ↑ 9000	28k	—	未 使 用
8FFF ↑ 8C00	1k	RAM	ユーザーズエリア (要拡張)
8BFF ↑ 8800	1k	RAM	ユーザーズエリア (要拡張)
87FF ↑ 8400	1k	RAM	ユーザーズエリア (要拡張)
83FF ↑ 83B1	55	RAM	モニタワーキングエリア
83C8 ↑ 83B1	24	RAM	RSTジャンプテーブル
83B0 ↑ 8391	32	RAM	モニタスタックエリア
8390 ↑ 8000	913	RAM	ユーザーズエリア
7FFF ↑ 2000	24k	—	未 使 用
1FFF ↑ 1800	2k	PROM	ユーザーズエリア (オプション)
17FF ↑ 1000	2k	PROM	ユーザーズエリア (オプション)
0FFF ↑ 0800	2k	PROM	ユーザーズエリア (オプション)
07FF ↑ 0000	2k	MASK ROM	モニタプログラム

3.2 RAMメモリマップ

アドレス	ラベル	シンボル	名称	名 考
83FF		DISP DIG 1		
FE		" 2		
FD		" 3		
FC		" 4	セグメントデータ	DMA転送によって、常時このエリアのセグメント
FB		" 5	バッファ	データが、LED上に表示されている
FA		" 6		
F9		" 7		
F8	DIG	" 8		
F7		DISP WORD 4		
F6		" 3	ディスプレイ	LEDディスプレイに表示するための16進データを
F5		" 2	レジスタ	セットするレジスタ
F4	DISP	" 1		
F3		(H)		
F2	SVBD	BREAK DEPTH (L)	ブレイクデプス	ブレイク動作時に、ループ回数をセットするレジスタ
F1		(H)		
F0	SVBP	BREAK POINT (L)	ブレイクポイント	ブレイク番地をセットするレジスタ
EF		(H)		
EE	ADRES	ADDRESS REG (L)	アドレスレジスタ	モニタがメモリに対して処理を行うときに参照するレジスタ
ED		(H)		
EC	DATA	DATA REG (L)	データレジスタ	ディジットキーより入力されたデータおよびメモリよりリードされたデータがセットされるレジスタ
EB		A		
EA	SVAF	F		
E9		B		
E8	SVBC	C		
E7		D		
E6	SVDE	E	CPUレジスタ	割り込みがなかった時点における、CPUレジスタ
E5		H	セーブエリア	の内容を退避するエリア
E4	SVHL	L		
E3		(H)		
E2	SVSP	S P (L)		
E1		(H)		
E0	SVPC	P C (L)		
DF	FMODE	MODE FLAG		
DE	FREG	REGISTER FLAG		
DD	FKEY	KEY FLAG	フラグエリア	モニタ処理時の状態を示すフラグのエリア
DC	FTRAP	TRAP FLAG		
DB	ENTCT	ENTER COUNTER		
DA		(H)		
D9		(L)	データセーブエリア	モニタ処理における、一時的なデータ退避エリア
D8		(H)		
D7	SVDAT	SAVE DATA (L)		

8章 モニタの詳細な説明

アドレス	ラベル	シンボル	名称	備考
83D6 D5 D4 D3	ADW1 ADW2 ADW3 ADW4	ADDRESS DISP WORD 1 " " "	2 3 4 アドレスディスプレイレジスタ	アドレスディスプレイに表示するディスプレイレジスタデータをセットするレジスタ
D2 D1 D0 CF	DDW1 DDW2 DDW3 DDW4	DATA DISP WORD 1 " " "	2 3 4 データディスプレイレジスタ	データディスプレイに表示するディスプレイレジスタデータをセットするレジスタ
CE CD CC	PORTO GOOUT	C 9 OUT PORT ADDRESS D 3	OUT命令エリア	OUT命令実行用エリア
CB CA C9	PORT1 GOIN	C 9 IN PORT ADDRESS D B	IN命令エリア	IN命令実行用エリア
C8 C7 C6		RST 7	RST 7ジャンプテーブル	RST命令によりジャンプしてくるエリア。このエリアにユーザーがジャンプ命令をセットすれば、RST命令実行後、設定エリアにジャンプすることができる。
C5 C4 C3		RST 6	RST 6ジャンプテーブル	
C2 C1 C0		RST 5.5	RST 5.5ジャンプテーブル	
BF BE BD		RST 5	RST 5ジャンプテーブル	
BC BB BA		RST 4	RST 4ジャンプテーブル	
B9 B8 B7		RST 3	RST 3ジャンプテーブル	
B6 B5 B4		RST 2	RST 2ジャンプテーブル	
B3 B2 B1		RST 1	RST 1ジャンプテーブル	
83B0 8391		MONITORS STACK	モニタスタックエリア	モニタが使用するスタックエリア
8390	USESP	USERS STACK ↓	ユーザースタック エリア ユーザエリア	ユーザーが使用できるRAMエリア。なおユーザープログラムを実行すると、8391番地にスタックポインタが設定される。
8000				

4. モニタを理解するために

3章の操作方法のところを読んで、モニタがどのような処理を行っているか理解できたことと思います。操作方法が理解できたということは、いわゆる外から見たモニタを理解したわけです。これではまだモニタ内部がブラックボックスのままで、操作は分かっても実際にどのようなアルゴリズムになっているかは依然未知のままです。このセクション以降はモニタ内部を理解するための説明です。

TK-85 モニタの構成は、通常のシステムプログラムの構成とは多少異なっています。通常のシステムプログラムは、一度各ファンクションに分岐したら、各ファンクションの動作が終了するまで多少の例外を除いて、そのシステムプログラムの先頭には戻りません。

ところが TK-85 のモニタは、入力キー一つに対する処理を行ったら、再びモニタの先頭に戻り、同じキー入力ルーチンでキースキャンを行っています。このようなモニタ形式をとった場合、問題になる点があります。それは一つのキーにいくつかの意味を持たせた場合、押されたキーが一体どの意味で押されたものなのか分からないということです。これが、通常のシステムプログラムのように、各ファンクション内がシーケンシャル構造となっている場合は、各ファンクション内におけるキーの意味付けが確定しているのです。このような問題は起こりません。

そこで TK-85 モニタでは、いくつかのフラグを設け、それを参照することによって、入力キーの解釈を行っています。

5. 分岐とフラグ

ワーキングエリアの中身を見ると、フラグエリアという領域が5バイト分確保されています。それぞれのフラグはプログラムの分岐用に設けられたものです。

まず各フラグを紹介し、それらのフラグがどのような用途に用いられているか、プログラムの流れと共に説明していきましょう。


5.1 モードフラグ (FMODE: 83DF 番地)

このフラグ構成は下図のようになっています。

7	6	5	4	3	2	1	0
IN	OUT	MOV	TM	LOAD	SAVE	未使用	MODE








操作方法の章でも述べましたが、モニタは三つのファンクションに分かれています。すなわち、ノーマルファンクション、モードファンクションそしてレジスタファンクションです。



このモードフラグはモードファンクション内部で用いられるフラグです。

 キーが押されると、アドレスディスプレイに4ドットのみを表示し、モードファンクションに入ったことを知らせます。

このとき、モニタはモードフラグの MODE をセットし、レジスタフラグおよびエンターカウンタをクリアします。そしてアドレスおよびデータディスプレイレジスタに BLANK のデータを格納し（アドレスおよびデータレジスタには 0 を格納）セグメントデータ変換を行い、再びキー入力待ちになります。

アドレスおよびデータディスプレイレジスタには BLANK のデータが格納されましたが、アドレスディスプレイ上には四つのドットが表示されています。これは、セグメントデータ変換サブルーチン内部でフラグセンスを行い、モードフラグの MODE ビットが立っていたときには、セグメントデータとドットのデータとの論理和をとっているからです。


 キーが押され、モードフラグの MODE ビットが立つと、キーボードスイッチのディジットモードファンクションキー（     ）のモードファンクション側が有効になります。


なお、このフラグビットが立っていると、モニタは   モードファンクションキーおよびハードウェアスイッチ以外のキー入力を受け付けません。

現在、アドレスディスプレイ上に四つのドットが表示され、モードの選択が要求されています。ここで各モードファンクションキーを入力することにより六つのモードのいずれかを選択することができます。

このとき、モニタは各モードファンクションに対応するフラグビットをセットし、MODE ビットをリセットします。その後セグメントデータ変換を行い、再びキー入力待ちになります。

IN モード、OUT モード以外はディスプレイレジスタへの操作を何も施していませんが、アドレスディスプレイ上には各モードに対応するドットが一つ表示され、どのモード内にいるのか一目で分かるようになっていました。これも先程と同様にドットの表示がすべて、セグメントデータ変換ルーチン内のフラグセンスによって制御されているからです。

この結果、モニタは各モードファンクションキーのモードになり、そのモードが終了するか、あるいはディジットキーおよび  キーを除いたキーが入力されるまでは、モードフラグの対応ビットが立ち続け、各モード処理内部にいる訳です。

なおモニタが各モード処理内部にいる場合は、 キーは ENTER キーとして作用します。フラグエリア内部に ENTCT が 1 バイト確保され、各モード処理内部で押される ENTER キーのカウンタを行っています。これは各モード処理内部で行われる処理が、入力 ENTER キーひとつによって異なっているためです。

5.2 レジスタフラグ (FREG: 83DE番地)

このフラグ構成は下図のようになっています。

7	6	5	4	3	2	1	0
BR.D	BR.P	SP	HL	DE	BC	AF	REG

このレジスタフラグはレジスタファンクション内部で用いられるフラグです。

REG キーが押されると、ディスプレイに、REG に対応するドットのみを表示し、レジスタファンクションに入ったことを知らせます。

このときモニタはレジスタフラグの REG をセットし、モードフラグおよびエンターカウンタをクリアします。そしてアドレスおよびデータディスプレイレジスタに BLANK のデータを格納し、(アドレスおよびデータレジスタには 0 を格納)、セグメントデータ変換を行い、再びキー入力待ちになります。

アドレスおよびデータディスプレイレジスタには BLANK のデータが格納されましたが、ディスプレイ上には一つのドットが表示されています。これもモードフラグの場合と同様にセグメントデータ変換ルーチン内で、レジスタフラグをセンスしてドット制御を行っているからです。

REG キーが押され、レジスタフラグの REG ビットが立つと、キーボードスイッチのディジットレジスタファンクションキー (**0 AF** **1 BC** **2 DE** **3 HL** **4 SP** **5 BP** **6 SI**) のレジスタファンクション側が有効になります。

なお、このフラグビットが立っていると、モニタは **MODE** **REG** レジスタファンクションキーおよびハードウェアスイッチ以外のキー入力を受け付けません。

現在、ディスプレイ上に一つのドットが表示され、表示レジスタの選択が要求されています。ここで各レジスタファンクションキーを入力することにより、五つのベアレジスタとブレイクポイント、そしてブレイクデプスの表示を選択することができます。

このときモニタは各レジスタに対応するビットをセットし、REG ビットをリセットします。その後選択されたレジスタの名前およびその内容を表示し、再びキー入力待ちになります。

レジスタ表示機能だけならばレジスタフラグの REG ビットのみで十分です。しかし、レジスタファンクションは単にレジスタの内容を表示するだけではなく、その内容を修正したり連続して他のレジスタの内容を表示させる機能を有しています。したがって、現在対象としているレジスタのポイントを設定する必要があります。レジスタフラグの REG を除く各ビットはこのポイントとしての役割を担っているのです。

5.3 トラップフラグ (FTRAP: 83DC 番地)

MON キーを押すと、TRAP 割り込みがかかります。

TRAP 割り込みは、インターラプトイネーブルや、割り込みマスクに関係なく割り込みがかかります。したがってモニタ内部でも割り込みがかかってしまい、**MON** キーがモニタ内部で押されたのか、ユーザーのアプリケーションプログラム内部で押されたのか見極める必要があります。トラップフラグはこれらの判断の手段として設けられたフラグです。

トラップフラグはモニタ内部では 00H (SYSTEM) という値をとり、ユーザープログラム内部では FFH (USER) という値をとります。

0 番地スタート、TRAP 割り込み処理ルーチン突入時および RST 7.5 割り込み処理ルーチン

突入時等、モニタプログラムへの入口でトラップフラグは00Hにセットされます。

またユーザープログラムへのジャンプ時、つまりモニタプログラムの出口でトラップフラグはFFHにセットされます。

なお、トラップフラグにともなう処理の説明は6節を参照して下さい。

5.4 キーフラグ (FKEY: 83DD 番地)

新たなキー入力を検出するために、キーフラグが設けられています。

キーフラグは、キーボードスイッチが押されている状態ではFFHという値をとり、キーボードスイッチが離されている状態で00Hという値をとります。

なお詳細な説明は10章4節の入力用サブルーチンの項を参照して下さい。

6. 割り込み処理とイニシャライズ

8085Aには5本の割り込み入力 (INTR, RST 5.5, RST 6.5, RST 7.5そしてTRAP) があります。TK-85では、RST 5.5を除く他の割り込み入力を使用しています。これらの割り込みに対してモニタがどのような処理を行っているのか、また電源投入時あるいはリセットボタンが押されたとき、どのような処理を行っているのか順を追って説明していきましょう。

6.1 0番地スタート

電源投入時、リセットボタンを押したとき、RST 0を実行させたとき、これらの動作が行われたときにはいずれも0番地から、モニタがスタートします。0番地からモニタがスタートすると、まず、モニタの使用しているRAMエリアつまりワーキングエリアのテストメモリを行います。ワーキングエリアのRAMが異常な場合は、HLT命令を実行し、CPUを止めてしましますが、正常であることが確認できた場合は、次のようなイニシャライズを行います。

まず、8255の動作モードをPORT A, PORT Bを入力、PORT Cを出力に指定します。次に83C9H番地から83F7H番地までを0クリアし、83C9H番地から83CEH番地までのIN命令、OUT命令エリアにそれぞれのインストラクションコードおよびRETコードを格納します。また、CPUレジスタセーブエリアのうち、スタックポインタのセーブエリアに最初のユーザースタックとなる番地8391Hを格納します。

以上のイニシャライズが終了すると、0クリアされているディスプレイレジスタの内容を、LEDディスプレイに表示し、キー入力待ちになります。

6.2 TRAP 割り込み処理

キーボードスイッチの一つに ☐ MON キーがあります。このキーを押すとモニタのWARM STARTを行うことができます。この ☐ MON キーの処理にはTRAP 割り込みを使用しています。

TRAP 割り込みは、他の割り込みがイネーブル状態であろうとなかろうと、それには無関係に割り込みがかかります。したがって ☐ MON キーを押せば、モニタ内部でもユーザープログラム内部でも割り込みがかかります。そこでモニタではTRAP 割り込みの処理を次のようにしています。

トラップフラグの項でも述べましたが、CPU の処理がモニタ内部なのか、ユーザープログラム内部なのかを判断するのにトラップフラグを用いています。その結果ユーザープログラム実行中に **MON** キーが押されたと判断した場合は、**MON** キーが押されたときの CPU レジスタの内容を CPU レジスタセーブエリアに退避します。次に 8255 の動作モードを再設定し、83C9H 番地から 83F7H 番地までの CPU レジスタセーブエリア以外のワーキングエリアをイニシャライズします。また、退避されたプログラムカウンタおよび AF の表示処理を行い、キー入力ルーチンに入ります。つまりユーザープログラム実行中に **MON** キーを押せばモニタの WARM START を行うことができます。

モニタ内部で **MON** キーが押されると、CPU レジスタの退避は行わず、8255 の動作モードの再設定および 83C9H 番地から 83F7H 番地までの CPU レジスタセーブエリア以外のワーキングエリアをイニシャライズし、0 クリアされているディスプレイレジスタの内容を LED ディスプレイに表示し、キー入力待ちになります。

ユーザープログラム暴走中 WARM START を行くと、スタックポインタが異常な値になることが往々にしてあります。WARM START を行ったときはスタックポインタを確認し、もしそれが異常な値になっている場合は、常にスタックポインタを修正し直すように心がけて下さい。

モニタはモニタスタックエリアのほかに、一部ユーザースタックエリアを使用しています。したがってスタックポインタが異常な値のままですとモニタが正常に動かない場合があります。

6.3 RST 7.5 割り込み処理

ステップスイッチを STEP 側に倒すと、割り込みイネーブル状態(ユーザープログラム実行中)のとき、1 命令実行する毎に RST 7.5 割り込みがかかります。割り込みがかかった後の処理は、ブレイクポイント、ブレイクデプスの内容により異なっています。

RST 7.5 の割り込み処理ルーチンに入ると、まず CPU レジスタの退避を行います。次にブレイクデプスの内容を調べ、それが 0 の場合はプログラムカウンタおよび AF を表示し、モニタのキー入力ルーチンに入ります。ブレイクデプスが 0 でない場合は、ブレイクポイントとプログラムカウンタの比較を行います。両者が一致しない場合はブレイクデプスをデクリメントし、再び 0 との比較を行います。0 のときは先程と同様にプログラムカウンタおよび AF を表示し、モニタのキー入力ルーチンに入ります。0 でないときは退避したレジスタの内容を復帰し再びユーザープログラムを実行します。

ブレイク動作を行いながらユーザープログラムを実行し、その途中で **MON** キーを押すと、PC、AF を表示せずに 0 を表示します。ブレイク動作を行うときはユーザープログラムを 1 命令実行する毎に RST 7.5 の割り込みがかかります。そのたびにモニタ処理を行っています。したがって一見ユーザープログラムのみを実行しているようですが、実際はモニタ処理がその大半を占めているのです。TRAP 処理で述べましたが、CPU がモニタを実行しているか、ユーザープログラムを実行しているかはトラップフラグによって判断しています。RST 7.5 の割り込みがかかると、ト

ラップフラグをモニタ側にセットしますが、トラップフラグをモニタ側にセットするまでにいくつかのインストラクションを実行しています。したがってブレイク動作を行っているときに TRAP 割り込みをかけた場合、モニタ内部のアドレスを表示する可能性もあります。

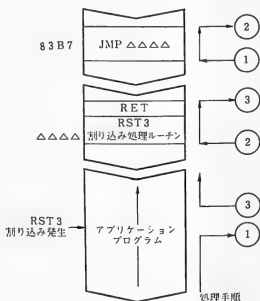
6.4 RST 6.5 割り込み処理

TK-85 では直接この割り込みを使用していません。RST 6.5 はTK-80BS のブレイク処理に用いられています。したがってモニタは、RST 6.5 の割り込みがかかったとき、TK-80BS のブレイクエントリーエリア、F125H番地にジャンプするようにしています。

6.5 リスタートジャンプテーブル

TK-85 モニタにおいて7種類のリスタート命令および1本の割り込み入力を開放しています。これらのリスタート命令を実行すると、おのおの次に示す番地に無条件にジャンプしていきます。したがってこのエリアに各処理ルーチンへのジャンプ命令を書き込んでおくことにより各処理を実行することができます。

RST 1	83B1H番地
RST 2	83B4H番地
RST 3	83B7H番地
RST 4	83BAH番地
RST 5	83BDH番地
RST 6	83C3H番地
RST 7	83C6H番地
RST 5.5	83C0H番地



7. モードファンクションの説明

7.1 IN モード, OUT モード

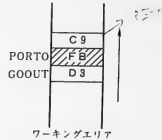
これら二つのモードファンクションにより、モニタ操作で直接ポートヘータを出力したり、ポートへの入力データを監視したりすることができます。

ワーキングエリアに IN 命令エリア, OUT 命令エリアがあり、モニタは直接ここに各インストラクションコード等を入れて、それぞれの命令を実行しています。しかしなぜこのようなことを行うのか疑問に思われる方がいるかもしれません。しかし、これにはそれなりの理由があるのです。

μ COM-85 あるいは μ COM-80 のインストラクション IN および OUT を参照して下さい。両者共オペランドはイミディエイトデータになっています。つまりポートアドレスを可変にすることができないのです。したがって IN および OUT 命令を実行するときには、IN および OUT ポートアドレスをワーキングエリアの RAM 上に格納することによって、ポートアドレスの可変化を図っているのです。

ポートアドレスが固定の場合 ポートアドレスが可変の場合

⋮	⋮
LDA DATE	MVI A, 0FBH
OUT 0FBH	STA PORT0
⋮	⋮
	LDA DATE
	CALL GOOUT
	⋮



7.2 MOVE モード

MOVE モードはデータのブロック転送を行います。その用途は実にバラエティに富んでいます。ここでは特にそのことには触れず、MOVE モードのアルゴリズムについて説明します。

データのブロック転送を行う場合、注意すべき事項があります。それはソースデータのエリアとディスティネーションデータのエリアが重なった場合、未転送のソースデータを破壊してしまう恐れがあるということです。図 8.2 の例を見て下さい。これは 8000H 番地から 8100H 番地までのデータ（ソースデータ）を 80C0H 番地以降に 8000H 番地から順に転送しようとしている図です。この場合、80C0H 番地から 8100 番地までのソースデータを破壊してしまい 8180H 番地から 81C0H 番地までのディスティネーションデータのエリアには、その破壊されたデータが格納されてしまいます。

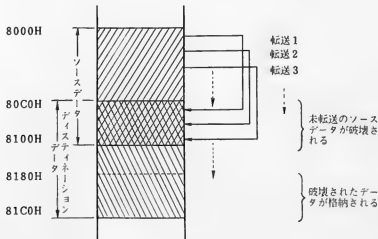


図 8.2

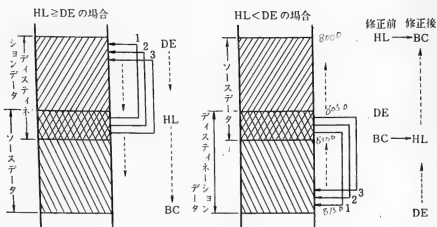


图 8.3

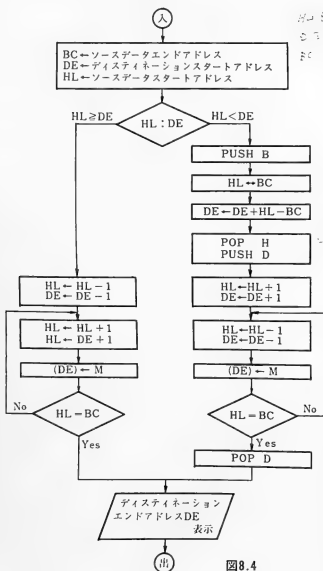


图 8.4

以上のような現象を避けるために、ソースデータとディスティネーションデータの先頭アドレスの比較を行い、その大小によって転送順序を変えています。

ソースデータの先頭アドレスが、ディスティネーションデータの先頭アドレスに等しいか大きいときは、ソースデータの先頭アドレスから END アドレスの方に向かってデータを転送していきます。またソースデータの先頭アドレスがディスティネーションデータの先頭アドレスより小さいときは、ソースデータの END アドレスから先頭アドレスの方に向かって転送していきます。

では、モニタがどのように処理しているか、転送部分のアルゴリズムを図 8.3 に示します。なお、HL レジスタにはソースデータの先頭アドレス、DE レジスタにはディスティネーションデータの先頭アドレス、BC レジスタにはソースデータの END アドレスが入っているものとします。

7.3 TEST MEMORY モード

テストメモリは、増設 RAM あるいは実装 RAM ユーザーエリアの READ WRITE 動作の確認およびアドレスバス、データバスの誤配線等のチェックを行うときに使用します。

実際のテストは、次のような書き込み動作と読み込み動作を繰り返しています。

まず指定されたテスト領域に先頭から順に 0, 1, 2, ……というデータを書き込みます。次に書き込みデータ、つまり期待値と、RAM からの読み込みデータとの比較を行い正常な READ WRITE 動作が行われたかどうかの確認を行います。正常な場合は、再びテスト領域に先頭から順に 1, 2, 3, ……というデータを書き込み、上記のようなチェック動作を繰り返します。この書き込み、読み込み動作はテストデータが 0 H から FFH まで計 256 回繰り返し行われます。

書き込み領域	書き込みデータ (HEXA)
	STAT → END
1	0, 1, 2, … FF, 0 …
2	1, 2, 3, ……………
3	2, 3, 4, ……………
4	3, 4, 5, ……………
.	.
.	.
.	.
255	FE, FF, 0 ……………
256	FF, 0, 1 ……………

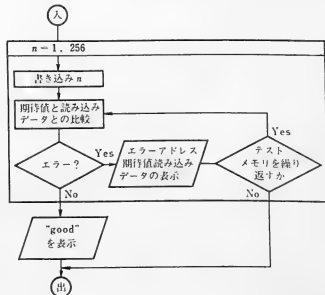


図 8.5

READ, WRITE の動作の確認およびデータバスの誤配線を調べるときは、RAM 領域 1 バイト毎に読み書きを行うことで動作確認は十分なのですが、アドレスバス誤配線があった場合、それだけではチェックすることができません。なぜなら異常アドレスに、テストデータを書き込んでも、その異常アドレスからデータを読み込んでしまうからです。したがって、全テスト領域にテストデータを書き込んでからチェックを始めるという方法をとっています。では実際にどうなるのか、図 8.6 のような 3 本のアドレス線からなる RAM を想定して説明しましょう。

3 本のアドレス線ですから、選択できるアドレスは 0 番地から 7 番地までです。ところが図 8.6 のような誤配線があった場合、2 番地から 5 番地までは、他のアドレスが選択されてしまいます。したがって 0 番地から 7 番地まで書き込み 1 を行うと実際には、0 番地に 4、1 番地に 5、6 番地に 6、7 番地に 7 というデータが書き込まれ、その他のアドレスには、書き込みが行われません。またこの後、0 番地から 7 番地までの読み込み動作を行うと、4、5、6、7、4、5、6、7 というデータが読み込まれてしまいます。書き込み 1 による期待値は 0、1、2、3、4、5、6、7 ですから、これでアドレスの誤配線が確認できるわけです。

テストメモリを行うと、その領域の RAM の内容を破壊してしまいます。したがってモニタワーキングエリアのテストメモリは行わないようにくれぐれも注意して下さい。なおワーキングエリアのテストメモリはモニタが走った時点ですでに行われています。

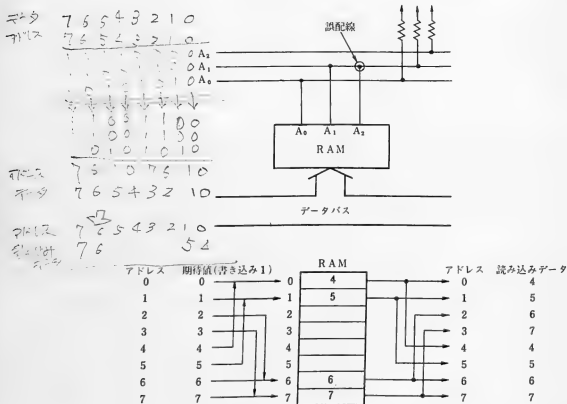


図 8.6

7.4 SAVE モード, LOAD モード

μPD8085A には SIO および SOD というシリアルデータ入出力用の端子があります。

カセットテープへのファイル作成およびカセットテープからのファイルロードはすべてこの端子を利用しています。

SIO 端子からの入力データは、RIM 命令によってアキュムレータのビット 7 にロードされます。

またアキュムレータのビット 6 (SOD ENABLE) がセットされているとき、SIM 命令によってアキュムレータのビット 7 が SOD 出力ラッチにロードされます。

データ入出力はこのようにシリアルデータを扱います。データは周波数 2400Hz の波 2 周期をデータ "1" とし、周波数 1200Hz の波 1 周期をデータ "0" としています。したがってデータの転送速度は 1200 baud (1200bit/s) になっています。

データは 1 バイトを一つの単位としています。1 バイトの転送データは、スタートビット ("0" × 1) ではじまり、ストップビット ("1" × 2) で終了します。したがって 1 バイト転送に要するビット数は 11、時間にして約 9.17ms です。

○ 1 k バイトデータ構成 (23H) : SOD 出力波形



図 8.7

1 バイトデータが前記のような形式になっていることが分かりました。カセットテープに生成させるファイルはこれら 1 バイトデータの集合である 1 プログラムが 1 単位となって構成されます。

各ファイルはファイル番号を有し、これによって各ファイルが判別されます。したがっていくつかのファイルがカセットテープにある場合も、これらファイル番号を基にファイル検索が可能になるわけです。

1 ファイルは、ファイルヘッド、アドレス部およびデータ部より構成されています。ファイルヘッドは READER 部および KEY WORD に分かれ、READER 部はデータ "1" 6000 ビットより成り、KEY WORD は 1 バイトデータ 55 H より成っています。

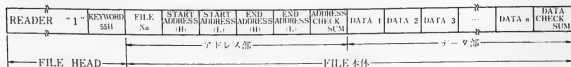
アドレス部は、1 バイトのファイル No., START ADDRESS HIGH, LOW および END ADDRESS HIGH, LOW そして、アドレスチェックサムより構成されています。1 バイトデータ構成によって気づかれたかと思いますが、1 バイトデータ中にパリティビットを設けておらずエラーがあっても認識できません。したがってアドレス部にエラーがあった場合、暴走によってワーキングエリア等を破壊してしまう恐れがあります。そこでアドレスエラー発生時点で LOAD モードを中断するためこのアドレスチェックサムを設けてあります。

8章 モニタの詳細な説明

データ部はその名の通り、1バイトデータの集まりです。このデータ部にもチェックサムを設けてあり、データロードが正常に行われたかどうかチェックしています。

なお詳しい波形生成方法および検出方法は10章のモニターサブルーチンを参照して下さい。

○ファイル構成



○ファイルヘッド



○ADDRESS CHEK SUM : アドレス部 (ADDRESS CHECK SUMを除く) の和の2の補数

○DATA CHECK SUM : データ部 (DATA CHECK SUMを除く) の和の2の補数

図8.8

8. μ PD8255 のプログラミング

μ PD8255 は μ PD8080A (μ PD8085A) マイクロコンピュータシステム用に開発されたプログラマブル汎用 I/O デバイスです。3 組 (8 ビット) の入出力ポートがあり、プログラムによる制御でデータ入力、データ出力、ステータス信号入力、コントロール信号出力に使用することができます。

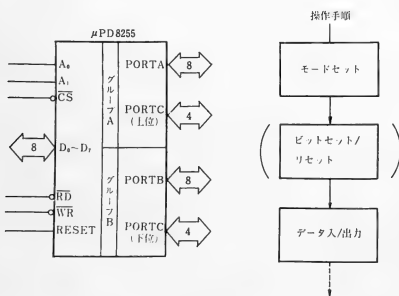


図8.9

μ PD8255 はその使用によって次の三つのモードに分けられます。

モード 0 : 基本的な入出力ポート

モード 1 : コントロール信号、ステータス信号による制御にともなう入出力ポート

モード 2 : 双方向データを扱う入出力ポート

モードの選択は CPU から μ PD8255 に対して送られる 8 ビットのコントロールワードによって行います。

ここでは最も基本的に使われるモード 0 での使い方を説明します (図 8.9 参照)。

8255 は RESET 信号を受けると、すべてのポートが入力モードに戻されて、以後命令で新しいモードがセットされない限り、この入力モードは継続されます。

モード 0 では基本的な入出力ポートとして動作します。CPU からコントロールワードを出力することによってポート A、ポート B、ポート C (上位)、ポート C (下位) がそれぞれ独立に、入力ポートあるいは出力ポートに設定されます。

コントロールワードを μ PD8255 に出力するときには、 μ PD8255 の各端子を次の状態にして出力しなければなりません。

\overline{CS}	A_0	A_1	\overline{RD}	\overline{WR}	動 作
0	1	1	1	0	データバス→制御 出力

ここで \overline{RD} および \overline{WR} はそのまま CPU の \overline{RD} および \overline{WR} に接続されているので入出力命令を使用するだけでそれぞれの機能を果し、特別ソフトウェアを組む上で考慮する必要はありません。

ここで問題になるのは \overline{CS} 、 A_1 、 A_0 です。8255 の A_1 および A_0 はアドレスバスの A_1 および A_0 にそれぞれ接続されています。 \overline{CS} は図 8.10 のようになっています。

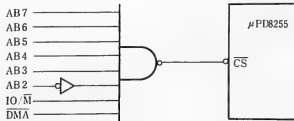


図8.10

これを見れば分かりますが、CS がアクティブになるのは次の条件が満たされたときです。まず DMA 転送時でないこと、リード/ライトが I/O に対して行われること、AB7～AB3 が High レベル ("1") AB2 が Low レベル ("0") であることです。

この中でソフトウェアが関与するのは、アドレスバスの部分だけです。したがって \overline{CS} をアクティブにするためには、アウトプットアドレスの内 AB7～AB3 を "1" AB2 を "0" に設定す

F B
7 6 5 4 3 2 1 0
1 1 1 1 0 1 1

ればよいわけです。

コントロールワードを μ PD8255に出力するときには上記の \overline{CS} をアクティブにする条件とA₁およびA₀を考慮すればいいので、アウトプットアドレスはFBH(AB7~AB3="1", AB2="0", AB1~AB0="1")となり、命令はOUT 0FBHとなります。

またこのアウトプットアドレスへの出力データつまりコントロールワードの仕様は図8.11のようになっています。

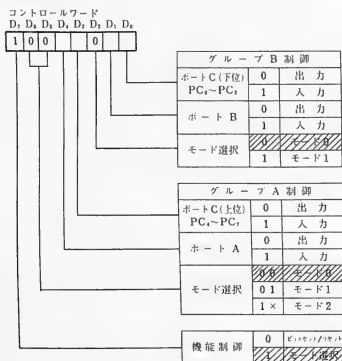


図8.11

TK-85 では、各ポートの入出力を次のようにしています。

ポート A : 入力 ポート B : 入力 ポート C : 出力

したがって μ PD8255へ送り込むコントロールワードは92Hとなり、TK-85のモニタの最初で次のような命令を実行しています。

MVI A, 92 H ; アキュムレータにコントロールワードをロードする。

OUT 0FBH ; μ PD8255 にコントロールワードを出力する。

コントロールワード送出後、各ポートとデータの入出力を行いますが、データを送り込む（アキュムレータの内容を各ポートへ送出）には OUT 命令を用い、各ポートからデータを入力する（各ポートからアキュムレータへロード）には IN 命令を用います。

このときの I/O アドレスはポートによって異なり、具体的には図8.12のようになります。

9. RIM, SIM について

コーディング		CS						A ₇	A ₆	RD	WR	動作	
		A ₇	A ₆	A ₅	A ₄	A ₃	A ₂						
IN	0F8H	1	1	1	1	1	0	0	0	0	1	ポート A→アキュムレータ	入 力
IN	0F9H	1	1	1	1	1	0	0	1	0	1	ポート B→アキュムレータ	
IN	0FAH	1	1	1	1	1	0	1	0	0	1	ポート C→アキュムレータ	
OUT	0F8H	1	1	1	1	1	0	0	0	1	0	アキュムレータ→ポート A	出 力
OUT	0F9H	1	1	1	1	1	0	0	1	1	0	アキュムレータ→ポート B	
OUT	0FAH	1	1	1	1	1	0	1	0	1	0	アキュムレータ→ポート C	

I/Q アドレス

図8.12

μPD8255 のポート C が出力としてプログラムされているときは、そのビットの位置を指定して、1 命令でセットまたはリセットすることができます。これは図 8.13 のようなコントロールワードを、μPD8255 に送り込む (OUT 0FBH) ことによって実現されます。

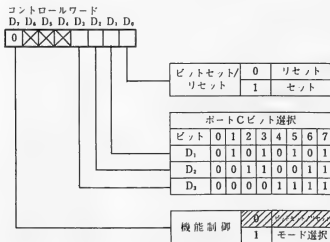


図8.13

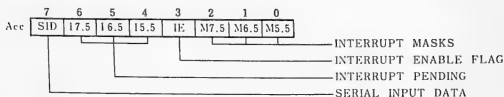
9. RIM, SIM について

μPD8085A には、μPD8080A に追加された命令が二つあります。それがこの RIM 命令と SIM 命令です。

RIM は read interrupt mask の略です。この命令を実行するとアキュムレータにリスタート割り込みマスク、保留割り込み、そして SID の内容がロードされます。

モニタはカセットテープからのシリアル入力に RIM 命令を用い SID 端子をシリアルデータの

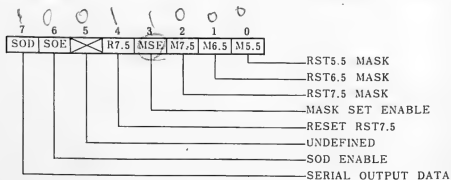
入力に使用しています。



SIM は set interrupt masks の略です。アキュムレータの内容がリスタート割り込みマスクのプログラミングのために用いられます。ビット 0～2 は、ビット 3 が 1 (セット) ならインターラプトマスクレジスタの RST 5.5, 6.5, 7.5 に対するマスクビットをセットまたはリセットします。ビット 3 は「マスクセット イネーブル」制御フラグです。マスクをセットすると (マスクビット = 1)、関連のインターラプトは禁止されます。RST 7.5 はマスクされてもされなくてもビット 4 = 1 ならセットされます。モニタは、ユーザープログラムにジャンプする時点ですべてのインターラプトマスクをリセットしています。

SIM は、SOD 出力ラッチのロードも行います。もしビット 6 がセットされていれば、アキュムレータのビット 7 が SOD 出力ラッチにロードされます。ビット 6 が 0 ならラッチは影響されません。

モニタはこの機能を、カセットテープへのシリアルデータ出力に使用しています。



9章 モニタプログラムリスト

E 51NO ADDR OBJECT M SOURCE STATEMENTS

```

0001                               TITLE      ' NEW TK-05  MONITOR'
0002                               *****
0003                               ***      NEW TK-05      **
0004                               ***      MONITOR      **
0005                               ***      '80.2.15  NEC      **
0006                               *****
0007                               |
0008                               |
0009                               *****
0010                               ***      MONITOR START      **
0011                               *****
0012                               |
0013 0000                               ORG      0
0014 0000 3E92                               RVI      A,CTRLW
0015 0002 03F8                               OUT      MODES
0016 0004 C35F00                            JMP      RESET
0017 0007                               ORG      0
0018 0008 C3B183                            JMP      RST1
0019 000B                               ORG      10H
0020 0010 C3B493                            JMP      RST2
0021 0013                               ORG      18H
0022 0018 C3B793                            JMP      RST3
0023 001B                               ORG      20H
0024 0020 C3BA93                            JMP      RST4
0025 0023                               ORG      24H
0026 0024 C3B301                            JMP      TRAP
0027 0027                               ORG      28H
0028 0028 C3BD93                            JMP      RST5
0029 002B                               ORG      2CH
0030 002C C3C093                            JMP      RST55
0031 002F                               ORG      30H
0032 0030 C3C393                            JMP      RST6
0033 0033                               ORG      34H
0034 0034 C325F1                            JMP      BSRBK
0035 0037                               ORG      38H
0036 0038 C3C493                            JMP      RST7
0037 003B                               ORG      3CH
0038 003C C31902                            JMP      STEP
0039                               *****
0040                               |
0041                               ***      INITIALIZE      **
0042                               *****
0043                               |
0044 003F 21C9B3                            CLEAR1: LXI      H,GOIN
0045 0042 AF                                CLEAR1: XRA      A
0046 0043 77                                CLEAR2: MOV      M,A
0047 0044 23                                INX      H
0048 0045 05                                DCR      B
0049 0046 C24300                            JNZ      CLEAR2
0050 0049 32DC83                            STA      FTRAP
0051 004C 3EC9                               RVI      A,OC9H
0052 004E 32CE83                            STA      PORT0+1
0053 0051 32CB83                            STA      PORT1+1
0054 0054 3ED3                               RVI      A,003H
0055 0056 32CC93                            STA      0007
0056 0059 3ED8                               RVI      A,00BH
0057 005B 32C993                            STA      GOIN
0058 005E C9                               RET
0059                               |
0060 005F 219183                            RESET: LXI      H,USESP
0061 0062 7E                                RESET1: MOV      A,H
0062 0063 2F                                CMA
0063 0064 77                                MOV      M,A
0064 0065 BE                                CMP      H
0065 0066 CA6A00                            JZ      RESE2
0066 0069 7E                                HALT
0067 006A 2C                                RESE2: INR      L
0068 006B C26200                            JNZ      RESE1
0069                               |
0070 006E 31B183                            LXI      SP,RST1
0071 0071 062F                            MVI      B,DIG-GOIN
0072 0073 C8F000                            CALL     CLEAR
0073 0076 219183                            LXI      H,USESP
0074 0079 22E283                            SHLD     SVSP
0075                               *****
0076                               |
0077                               *****
0078                               |
0079                               |
0080 007E CDE005                            START: CALL     SEBCV
0081 007F CDB406                            CALL     KEYIN
0082                               |
0083 0082 E610                            STAT1: ANI      10H
0084 0084 CAB800                            JZ      DIGIT

```

1TRANSFER CONTROL WORD (MODE 0) PAIRIN PB:IN PC:OUT) TO B255.
 1JUMP INITIALIZE AREA
 1JUMP RST1 PROCESSING AREA
 1JUMP RST2 PROCESSING AREA
 1JUMP RST3 PROCESSING AREA
 1JUMP RST4 PROCESSING AREA
 1JUMP MONITOR
 1JUMP RST5 PROCESSING AREA
 1JUMP RST5.5 PROCESSING AREA
 1JUMP RST6 PROCESSING AREA
 1JUMP BREAK ENTRY AREA (TK-80 B5)
 1JUMP RST7 PROCESSING AREA
 1JUMP BREAK ENTRY AREA (NEW TK-05)

1SET TOP ADD. OF CLEAR AREA
 1CLEAR WORKING AREA
 1SET TRAP FLAG TO SYSTEM (0)
 1INITIALIZE OUT INSTRUCTION AREA
 1INITIALIZE IN INSTRUCTION AREA
 1SET OUT INSTRUCTION CODE
 1SET IN INSTRUCTION CODE
 1SET TOP ADD. OF MONITOR'S WORKING AREA
 1IF WORKING AREA ERROR THEN HALT
 1SET TOP ADD. OF MONITOR'S STACK
 1SET CLEAR COUNTER
 1MOVE TOP ADD. OF USER'S STACK TO (SP)SAVING AREA
 1SEGMENT DATA CONVERSION
 1KEY INPUT (A-BIHEXA DATA)
 1IF INPUT DATA IS EQUAL TO DIGIT THEN JUMP DIGIT PROCESS

9章 モニタプログラムリスト

```

0085
0086 0087 3ADF83      LDA     FMODE
0087 0088 OF          RRC     STAT2
0088 0089 DAF500      JC      STAT2      ;IF F-MODE IS ON THEN SELECT INPUT DATA
0089
0090 0091 3ADE83      LDA     FREG
0091 0092 OF          RRC     STAT3
0092 0093 D29600      JNC     STAT3      ;IF F-REG IS ON THEN SELECT INPUT DATA
0093
0094 0095 78          STAT2: MOV     A,B
0095 0096 FE16        CPI     1&H
0096 0097 DAF700      JC      START      ;IF INPUT DATA IS EQUAL TO REG OR MODE THEN EXECUTE THIS ONE
0097
0098 0099 78          STAT3: MOV     A,B
0099 009C E64F        ANI     0FH
0100 009E 0400        MVI     B,0
0101 00A0 87          ADD     A
0102 00A1 4F          MOV     C,A
0103 00A2 21AB00      LXI     H,TFUNC
0104 00A5 09          DAD     B
0105 00A6 7E          MOV     A,M
0106 00A7 23          INX     H
0107 00AB 4A          MOV     H,M
0108 00A9 6F          MOV     L,A
0109 00AA E9          PCHL
0110
0111 00AB 9701      TFUNC: DW     RUN
0112 00AD A401      DW     CDW
0113 00AF 6A01      DW     ADDRST
0114 00B1 B001      DW     RDOEC
0115 00B3 8301      DW     RDINC
0116 00B5 7902      DW     WTEMP
0117 00B7 DA00      DW     MDPKP
0118 00B9 E900      DW     REG
0119
0120
0121 *****
0122 *** DIGIT PROCESS ***
0123 *****
0124 009B 3ADE83      DIGIT: LDA     FREG
0125 00BE OF          RRC     JC      REGFN      ;IF F-REG IS ON THEN DISPLAY REGISTER
0126 00BF DA1201      JC
0127
0128 00C2 3ADF83      LDA     FMODE
0129 00C5 OF          RRC     JC      MD      ;IF F-MODE IS ON THEN SET EACH MODE FUNCTION FLAG
0130 00C6 DAF200      JC
0131
0132 00C9 CDD205      CALL    SHIFT
0133 00CC 3AEC93      LDA     DATA
0134 00CF 80          DRA     B
0135 00D0 32C83      STA     DATA      ;INPUT DIGIT DATA (DR)
0136 00D3 78          MOV     A,B
0137 00D4 32CF83      STA     DOW4      ;INPUT DIGIT DATA (DOW)
0138 00D7 C37C00      JMP     START
0139
0140 *****
0141 *** FLAG(MODE&REG) PROCESS ***
0142 *****
0143
0144 00DA CDE905      MDPKP: CALL    FLAG      ;RESET EACH FLAG
0145 00DD 32DF83      STA     FMODE      ;SET F-MODE
0146 00E0 CD2906      ALCLA: CALL    CA      ;CLEAR (ADM) & (AR)
0147 00E3 CD3906      DTCLA: CALL    CD      ;CLEAR (DOW) & (DR)
0148 00E6 C37C00      JMP     START
0149
0150 00E9 CDE905      REG:  CALL    FLAG      ;RESET EACH FLAG
0151 00EC 32DF83      STA     FREG      ;SET F-REG
0152 00EF C3E000      JMP     ALCLA
0153
0154 *****
0155 *** MODE FUNCTION KEY PROCESS ***
0156 *****
0157
0158 00F2 78          MD:  MOV     A,B
0159 00F3 FE0A        CPI     0AH
0160 00F5 DA7C00      JC      START      ;IF INPUT DATA IS NOT EQUAL TO MODE FUNCTION THEN INPUT AGAIN
0161
0162 00F8 D608          SUJ     B
0163 00FA CDF305      CALL    STBIT      ;SET BIT
0164 00FD 32DF83      STA     FMODE      ;SET EACH MODE FLAG BIT (RESET F-MODE)
0165 0100 E640        ANI     00H
0166 0102 CA7C00      JZ      START      ;IF F-IN OR OUT IS ON THEN IN/OUT PROCESS
0167
0168 0105 17          RAL
0169 0106 17          RAL
0170 0107 32DA83      STA     ADM1      ;SET "0" OR "1"
0171 010A 3E1D        MVI     A,1D
0172 010C 32DF83      STA     RUNC2
0173 010F C37C00      JMP     START
0174
0175 *****
0176 *** DISPLAY REGISTER ***
0177 *****
0178
0179 0112 78          REGFN: MOV     A,B
0180 0113 FE07        CPI     7
0181 0115 D27C00      JNC     START      ;IF INPUT DATA IS GREATER THAN 6 THEN START
0182
0183 0118 CDF305      CALL    STBIT      ;SET BIT
0184 011B 07          RLC
0185 011C 32DE83      STA     FREG      ;SET EACH REGISTER FLAG BIT (RESET F-REG)
0186
0187 011F 78          MOV     A,B

```

```

0189 0120 0600      MVI      B,0
0189 0122 67        ADD      A
0190 0123 4F        MOV      C,A
0191 0124 214A01     LXI      H,TRGM
0192 0127 09        DAD      B
0193 0128 7E        MOV      A,M
0194 0129 23        INX      H
0195 012A 66        MOV      H,M
0196 012B 6F        MOV      L,A
0197 012C 2205E3     SHLD     ADM2      ;DISPLAY REGISTER NAME
0198
0199 012F 215601     ;
0200 0132 09        DAD      B
0201 0133 7E        MOV      A,M
0202 0134 23        INX      H
0203 0135 66        MOV      H,M      ;SET H,ADD. OF REGISTER SAVING AREA
0204 0136 6F        MOV      L,A      ;SET L,ADD. OF REGISTER SAVING AREA
0205 0137 7E        MOV      A,M
0206 0138 23        INX      H
0207 0139 66        MOV      H,M      ;SET (H,ADD. OF REGISTER SAVING AREA)
0208 013A 6F        MOV      L,A      ;SET (L,ADD. OF REGISTER SAVING AREA)
0209 013B 22ECB3     SHLD     DATA      ;SET (REGISTER SAVING AREA)
0210
0211 013E 211D1D     LXI      H,1D1DH
0212 0141 22D3B3     SHLD     DMCH
0213 0144 CB4A06     CALL     DMCH
0214 0147 C37C00     JMP      START
0215
0216 014A 0F0A      TRGMH: DW      0A0FH
0217 014C 0C0B      DW      0B0CH
0218 014E 0E0D      DW      0D0EH
0219 0150 1210      DW      1012H
0220 0152 1505      DW      0515H
0221 0154 150B      DW      0B15H
0222 0156 0D0B      DW      0B0DH
0223
0224 015B EA93      TROAD1: DW      SVAF
0225 015A EEB3      DW      SVBC
0226 015C EA93      DW      SVDE
0227 015E E4E3      DW      SVML
0228 0160 E293      DW      SVSP
0229 0162 F0B3      DW      SVNP
0230 0164 F293      DW      SVBD
0231
0232
0233 ;*****
0234 ;** ADDRESS SET PROCESS *****
0235 ;*****
0236 0166 D7C01     ;
0237 AERST1: CALL     ARSET      ;MOVE (DR) TO (AR)
0238 MEMRD1: MOV      A,M
0239 STA      DATA      ;READ MEMORY
0240 016D CD5106     CALL     ANCH      ;CONVERT (AR) INTO (ADM)
0241 0170 CD4306     CALL     CB12      ;CLEAR (DR)H & (DDW)1,2)
0242 0173 CD6D06     CALL     DLWCH      ;CONVERT (DR)L INTO (DDW)3,4)
0243 0176 CD6F05     CALL     FLAGC      ;RESET EACH FLAG
0244 0179 C37C00     JMP      START
0245
0246 017C 2AECB3     ;
0247 017F 22EEB3     ARSET1: LHD      DATA
0248 0182 C9        SHLD     ADRES
0249 RET
0250
0251 ;*****
0252 ;** READ INCREMENT PROCESS **
0253 ;*****
0254
0255 0183 2AE6B3     ;
0256 0186 23        RDINC1: LHD      ADRES
0257 0187 22EBB3     INX      H      ;INCREMENT (AR)
0258 018A C36901     SHLD     ADRES
0259 018D 23        JMP      MEMRD
0260
0261 ;*****
0262 ;** READ DECREMENT PROCESS **
0263 ;*****
0264
0265 018D 2AE6B3     ;
0266 0190 2B        RDDEC1: LHD      ADRES
0267 0193 22EBB3     DCR      H      ;DECREMENT (AR)
0268 0196 C36901     SHLD     ADRES
0269 0199 23        JMP      MEMRD
0270
0271 ;*****
0272 0197 2AE6B3     ;
0273 019A 22E0B3     RUN1: LHD      ADRES
0274 019D C34A01     SHLD     SVPC      ;MOVE (AR) TO (PC)SAVING AREA
0275 019F 23        JMP      CONT
0276
0277 ; <<<' TK-80 MONITOR SUBROUTINE "RGDSP" >>>
0278 01A0 ORG      01A1H
0279 01A1 C3CF07     JMP      RGDSP
0280
0281 ;*****
0282 ;** CONTINUE PROCESS **
0283 ;*****
0284
0285 01A4 2AE2B3     ;
0286 01A7 F9        CONT1: LHD      SVSP
0287 01AA 2AE0B3     SPHL     SVPC      ;SET ((SP)SAVING AREA)
0288 01AB E5        LHD      SVPC      ;SAVE ((PC)SAVING AREA)
0289 01AC 2AE6B3     PUSH     H
0290 01AF E5        LHD      SVAF      ;SAVE ((AF)SAVING AREA)
0291 01B0 E5        PUSH     H

```

9章 モニタプログラムリスト

```

0291 0180 2AE4B3      LHL D SVHL
0292 0183 E5          PUSH H
0293                  I
0294 0184 2AE093      LHL D SVBC
0295 0187 E5          PUSH H
0296 0189 C1          POP B
0297 0189 2AE6A3      LHL D SVDE
0298 018C EB          XCHG
0299 018D C3C301      JMP CONTJ
0300
0301                  I
0302                  I <<< TK-BO MONITOR SUBROUTINE "SEGC" >>>
0303 01C0
0304 01C0 C3DF07      ORG 01C0H
0305                  I
0306 01C3 E1          CONTJ: POP H
0307                  I
0308 01C4 AF          XRA A
0309 01C5 D3FA      OUT PORTC
0310 01C7 3D          DCR A
0311 01C8 32DCB3      STA FTRAP
0312 01CB 3E98      MVI A,RSKRS
0313 01CD 30          SIM
0314 01CE D3FA      OUT PORTC
0315                  I
0316 01D0 F1          POP PSW
0317 01D1 FB          EI
0318 01D2 C9          RET
0319                  I
0320 *****
0321 *** TRAP INTERRUPT PROCESS **
0322 *****
0323                  I
0324 01D3 F5          TRAP: PUSH PSW
0325 01D4 34BC93      LDA FTRAP
0326 01D7 A7          ANA A
0327 01DB C2E401      JNZ TRAPJ
0328                  I
0329 01DB 31B1B3      LXI SP,RSTJ
0330 01DE CD0202      CALL INIT
0331 01E1 C37C00      JMP START
0332                  I
0333 01E4 22E4B3      TRAPJ: SHLD SVHL
0334 01E7 210400      LXI H-4
0335 01EA 39          DAD SP
0336 01EB 22E2B3      SHLD SVSP
0337 01EE F1          POP PSW
0338 01EF E1          POP H
0339 01F0 22E0B3      SHLD SVPC
0340 01F3 31ECB3      LXI SP,DATA
0341 01FA F5          PUSH PSW
0342 01F7 E5          PUSH B
0343 01F8 D5          PUSH D
0344 01F9 31B1B3      LXI SP,RSTJ
0345 01FC CD0202      CALL INIT
0346 01FF C35D02      JMP BRET
0347                  I
0348 0202 3E92      INIT: MVI A,CTRLW
0349 0204 D3FB      OUT MODEM
0350 020A 0417      DCR B-SVPC-GOIN
0351 020E D3F000      CALL CLEAR
0352 020B 21ECB3      LXI H-DATA
0353 020E 060C      MVI B,DIG-DATA
0354 0210 C34200      JMP CLEAR
0355                  I
0356                  I <<< TK-BO MONITOR SUBROUTINE "KEYIN" >>>
0357                  I
0358 0213
0359 0216 C3406      ORG 0216H
0360                  I
0361                  I *****
0362 *** RST7.5 INTERRUPT PROCESS **
0363 *****
0364                  I
0365 0219 E3          STEP: XTHL
0366 021A 22E0B3      SHLD SVPC
0367 021D F5          PUSH PSW
0368 021E C32602      JMP STEPJ
0369                  I
0370                  I <<< TK-BO MONITOR SUBROUTINE "INPUT" >>>
0371                  I
0372 0221
0373 0223 C39006      ORG 0223H
0374                  I
0375 0226 210400      JMP INPUT
0376 0229 39          DAD SP
0377 022A 22E2B3      SHLD SVSP
0378 022D F1          POP PSW
0379 022E E1          POP H
0380 022F 31ECB3      LXI SP,DATA
0381 0232 F5          PUSH PSW
0382 0233 C3          PUSH B
0383 0234 D5          PUSH D
0384 0237 E5          PUSH H
0385 0236 31B1B3      LXI SP,RSTJ
0386                  I
0387 0239 AF          XRA A
0388 023A 32DCB3      STA FTRAP
0389 023D C3DF07      CALL BRDPT
0390 0240 C3D502      JZ BRET
0391                  I
0392 0243 34B0B3      LHL D SVB
0393 0246 EB          XCHG

```

```

0394 0247 2AE0E3      LMLD  SVPC      1HL : PC
0395 024A CDFE05      CALL  COMPA     1IF (PC) IS EQUAL TO (BP) THEN SET ZERO FLAG ELSE RESET ONE
0396 024D C2A401      JNZ   SVBD      1IF (PC) IS NOT EQUAL TO (BP) THEN CONTINUE
0397 0250 2AE2B4      LMLD  SVBD
0398 0253 2B          DCX   H          1IF (PC) IS EQUAL TO (BP) THEN DECREMENT (BD)
0399 0254 22F2B3      SHLD  SVBD
0400 0257 1AEF02      CALL  BREFCT
0401 025A C2A401      JNZ   CONT      1IF (BD) IS NOT EQUAL TO 0 THEN CONTINUE
0402
0403 025B 2AE093      BRET: LMLD  SVPC
0404 0260 22FE83      SHLD  ADRES     1MOVE (PC) TO (AR)
0405 0263 2AEAB3      LMLD  SVAF
0406 0266 22FE83      CALL  CH
0407 0269 CDE06        CALL  CH
0408 026C C37C00      JMP   START
0409
0410 0264 1AE2B3      RREG: LDA  SVBD
0411 0272 A7          ANA   A
0412 0275 C0          RNZ
0413 027A 5AFB93      LDA  SVBD+1
0414 0277 A7          ANA   A
0415 027B C9          RET
0416
0417
0418
0419
0420
0421 0279 3ADF83      WENT: LNA   FMODE
0422 027C A7          ANA   A
0423 027D C2BF02      JNZ   FPMOD     1IF MODE FLAG IS ON THEN JUMP EACH MODE FUNCTION PROCESS
0424
0425 0280 3ADEB3      LDA   FREG
0426 0283 A7          ANA   A
0427 0284 C2F502      JNZ   PRREG     1IF REG FLAG IS ON THEN MODIFY REGISTER AND DISPLAY NEXT ONE
0428
0429
0430 0287 2AE893      LMLD  ADRES     1WRITE MEMORY
0431 028A 2AE893      LDA   DATA     1READ AFTER WRITE
0432 028E 8E          MOV   M,A
0433 0291 C21706      CPM   M
0434 0292 C39301      JNZ   ERR       1IF WE COULD NOT WRITE DATA TO MEMORY THEN DISPLAY E-MESSAGE
0435
0436
0437
0438
0439
0440 0295 215E01      PRREG: LXI   H,TRGAD 1SET TOP ADD. OF SAVING REGISTER ADD.'S TABLE
0441 0298 0601      MVI   B,1
0442 029A 3AECE3      LDA   FREG
0443 029D 0F      RRC       1SET REGISTER FLAG
0444 029E 0F      RRC
0445 029F DAE802      RST   JC       1IF EACH REGISTER FLAG BIT IS ON THEN MODIFY ITS DATA
0446 02A2 23      INX   H
0447 02A3 23      INX   H
0448 02A4 04      INR   B
0449 02A5 C39E02      JMP   RSLFT
0450
0451 02A6 7E      DISET: MOV  A,M
0452 02A9 23      INX   H
0453 02AA 66      MOV   M,M
0454 02AB AF      MOV   L,A
0455 02AC 3AECB3      LDA   DATA
0456 02AF 77      MOV   M,A      1MODIFY (REGISTER SAVING AREA)L
0457 02B0 23      INX   H
0458 02B1 3AED93      LDA   DATA+1
0459 02B4 77      MOV   M,A      1MODIFY (REGISTER SAVING AREA)M
0460 02B5 79      MOV   A,B
0461 02B6 EE07      RRI   7
0462 02B8 C31201      JNZ   REGFN
0463 02BB 47      MOV   B,A
0464 02BC C31201      JMP   REGFN      1IF B IS EQUAL TO 7 THEN INITIALIZE B BECAUSE OF REPEATING
0465
0466
0467
0468
0469
0470 02BF 07      FRMOD: RLC
0471 02C0 DAD602      JC   FIN
0472 02C3 07      RLC
0473 02C4 DAF002      JC   FOUT
0474 02C7 07      RLC
0475 02C8 DAD203      JC   FMOV
0476 02CB 07      RLC
0477 02CC DAD203      JC   FTH
0478 02CF 07      RLC
0479 02D0 DAD004      JC   FLOAD
0480 02D3 C32504      JMP   FSAVE      1JUMP SAVE PROCESS
0481
0482
0483
0484
0485
0486 02D6 3ADB03      FIN:  LDA  ENTCT
0487 02D9 A7          ANA   A
0488 02DA C2ED02      JNZ   PJN
0489
0490 02DD 3C          INR   A
0491 02DE 32DB83      STA  ENTCT
0492 02E1 3AECB3      LDA  DATA
0493 02E4 32CA93      STA  PORT1
0494 02E7 D7706      CALL  MORDC
0495 02EA 220393      SHLD  ADH4
0496

```

9章 モニタプログラムリスト

```

0497 02ED CDC9B3      PINI: CALL    B0IN      ;EXECUTE IN INSTRUCTION
0499 02F0 32C0B3      STA     DATA    ;SET INPUT DATA
0499 02F3 CD7706      CALL    WORDC
0500 02F6 22CFB3      SHLD    DDW4      ;SET INPUT DATA
0501 02F9 CD4306      OUTINI: CALL   CD12
0502 02FC C37C00      JMP     START      ;CLEAR (DDW1,2) & (DRH)
0503
0504
0505
0506
0507
0508 02FF 3AD8B3      ;
0509 0302 A7          FOUT:  LDA     ENTCT
0510 0303 C21903      JNZ     POUT      ;IF ENTCT IS NOT EQUAL TO 0 THEN EXECUTE OUT INSTRUCTION
0511
0512 0306 3C          ;
0513 0307 32DBB3      INR     A
0514 030A 3AECB3      STA     ENTCT    ;INCREMENT ENTCT
0515 030D 32CDB3      LDA     DATA
0516 0310 CD7706      STA     PORTO    ;SET OUT PORT ADD.
0517 0313 22DBB3      CALL    WORDC
0518 0316 C3E300      SHLD    ADW4      ;SET OUT PORT ADD.
0519      JMP     DTCLA
0520 0319 3AECB3      ;
0521 031C CDC0B3      POUT:  LDA     DATA    ;SET OUT DATA
0522 031F C3F902      CALL    GOOUT    ;EXECUTE OUT INSTRUCTION
0523      JMP     OUTINI
0524
0525
0526
0527
0528 0322 3AD8B3      ;
0529 0325 A7          FTM1:  LDA     ENTCT
0530 0326 C23903      ANA     A
0531      JNZ     FTM1      ;IF ENTCT IS NOT EQUAL TO 0 THEN FTM1
0532 0329 3C          ;
0533 032A 32DBB3      INR     A
0534 032D CD7706      STA     ENTCT    ;INCREMENT ENTCT
0535 0330 22DBB3      CALL    ARSET    ;MOVE (DR) TO (AR)
0536 0333 CD5106      SVDAT   SVDAT      ;SAVE START ADD.
0537 0336 C3E300      CALL    AMCH      ;CONVERT (AR) INTO (ADW)
0538      JMP     DTCLA
0539
0539 0339 AF          FTM1:  XRA     A
0540 033A 4F          MOV     C,A
0541 033B 2AECB3      LHL     DATA
0542 033E EB          XCHG
0543
0544
0544 033F 2AD7B3      FTM2:  LHL     SVDAT
0545 0342 CD1206      CALL    ERRMS
0546 0345 41          B,C
0547 0346 28          DCX     H
0548 0347 25          INX     H
0549 0348 70          MOV     M,B
0550 0349 04          B
0551 034A CDF005      CALL    COMPA
0552 034D C24703      JNZ     FTM3
0553
0554
0554 0350 41          ;
0555 0351 2AD7B3      MOV     B,C
0556 0354 2B          LHL     SVDAT
0557 0355 23          DCX     H
0558 0356 78          MOV     A,B
0559 0357 BE          CMP     M
0560 0358 C27B03      JNZ     ERROR
0561 035B 04          MOV     A,B
0562 035C CDF005      CALL    COMPA
0563 035F C25503      JNZ     FTM4
0564 0362 0C          INR     C
0565 0363 C23F03      JNZ     FTM2
0566
0567 0366 CD2906      ;
0568 0369 211409      CALL    EA
0569 036C 2201B3      LXI     H,914H
0570 036F 210B14      SHLD    DDW2
0571 0372 22CFB3      LXI     H,140DH
0572 0375 CDE905      SHLD    DDW4
0573 0378 C37C00      CALL    FLAGC
0574      JMP     START
0575
0575 037B 22EEB3      ;
0576 037E 60          ERROR: SHLD    ADRES
0577 037F 6E          MOV     H,B
0578 0380 22EC33      MOV     L,M
0579 0383 C5          SHLD    DATA
0580 0384 D5          PUSH   B
0581 0385 E5          PUSH   D
0582 0386 CD4E06      PUSH   H
0583 0389 CD4C05      CALL    CH
0584 038C CD3406      CALL    SEGCV
0585 038F E1          CALL    KEYIN
0586 0390 D1          POP     H
0587 0391 C1          POP     D
0588 0392 FE15      POP     B
0589 0394 2AEEB3      CPI     15H
0590 0397 CA5B03      LHL     ADRES
0591 039A 47          JZ     FTM5
0592 039B CDE905      MOV     B,A
0593 039E 78          CALL    FLAGC
0594 039F C38200      MOV     A,B
0595      JMP     STAT1
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999

```



```

0000 03A2 3AD0B3  FMOV1 LDA ENTCT
0001 03A5 A7 ANA A
0002 03A6 C7BC04 JNZ FMOV1 ;IF ENTCT IS GREATER THAN 0 THEN FMOV1
0003 03A7 3C INR A
0004 03A9 3E STA ENTCT ;INCREMENT ENTCT
0005 03AA 32B014 CALL ARSET ;MOVE (IR) TO (AR)
0006 03AB 1D7C01 CALL ARCH ;CONVERT (AR) INTO (ADW)
0007 03AC CD5106 CALL DTCLA
0008 03AD 4A5001 JMP
0009 03AE 08
0010 03AF 08 FMOV11 RRC ;IF ENTCT IS GREATER THAN 1 THEN FMOV11
0011 03B0 D6C03 JNC ;A-2
0012 03B1 80 STA ENTCT ;INCREMENT ENTCT
0013 03B2 42B0F3 MVI DATA
0014 03B3 2A0C14 LHL SVDAT
0015 03B4 22D703 XCHG
0016 03B5 2A0F1E LHL ADRES
0017 03B6 2A0F1E LHL SVDAT+2
0018 03B7 197 22D703 JAL ERMS
0019 03B8 191206 CALL ERMS
0020 03B9 0A0000 JMP ALCLA
0021 03C0
0022 03C1 03C0 03C0 CALL ARSET ;MOVE (R) TO (AR)
0023 03C2 87 PUSH H
0024 03C3 1D0106 CALL ARCH ;SET START ADD. OF DESTINATION DATA
0025 03C4 03C7 CD9306 CD ;CLEAR (DDW) & (DR)
0026 03C5 D6A005 CALL SEGCV ;SEGMENT DATA CONVERSION
0027 03C6 2AD793 LHL SVDAT
0028 03C7 E5 PUSH H
0029 03C8 13 POP B
0030 03C9 D1 POP D
0031 03CA 2AD793 LHL SVDAT+2 ;BC : END ADD. OF SOURCE DATA
;DE : START ADD. OF DESTINATION DATA
;HL : START ADD. OF SOURCE DATA
0032 03CB
0033 03CC CDFF05 CALL COMPA
0034 03CD D20704 JNC BTUP1 ;IF HL IS LESS THAN OR EQUAL TO DE THEN EXECUTE BOTTOM UP
;IF HL IS GREATER THAN DE THEN EXECUTE TOP D-WN
0035 03CE 2B DCX H
0036 03CF 1B DCX D
0037 03D0 23 INX H
0038 03D1 13 INX D
0039 03D2 7E MOV A,M
0040 03D3 12 STAX D
0041 03D4 13 CALL CHLBC ;MOVE DATA
;IF START ADD. IS NOT EQUAL TO END ADD. THEN MOVE AGAIN
0042 03D5 4C0C06 JNZ TPDW1
0043 03D6 C2F003
0044 03D7
0045 03D8 03FA EB DSPDE1 XCHG
0046 03D9 22E103 SHLD DATA
0047 03DA CD6A06 CALL DMCH ;SET END ADD. OF DESTINATION DATA
0048 03DB CDE005 CALL FLAGC ;CONVERT (DR) INTO (DDW)
0049 03DC 0404 C37C00 JMP START ;RESET EACH FLAG
0050 03DD
0051 03DE 0407 E5 BTUP11 PUSH B ;SAVE HL (BC)
0052 03DF 0408 C5 PUSH B
0053 03E0 0409 E5 PUSH H
0054 03E1 040A C1 POP B
0055 03E2 040B E1 POP B
0056 03E3 040C 19 POP H
0057 03E4 040D 7D DAD D
0058 03E5 040E 71 MOV A,L
0059 03E6 040F 53 SUB C
0060 03E7 0410 7C MOV E,A
0061 03E8 0411 9B MOV A,H
0062 03E9 0412 57 SBB B
0063 03EA 0413 E1 MOV D,A ;CALCULATE (HL+DE-BC) AND SET ITS SOLUTION
0064 03EB 0414 05 POP H
0065 03EC 0415 23 PUSH D
0066 03ED 0416 13 INX H
0067 03EE 0417 2B INX D
0068 03EF 0418 1B DCX H
0069 03F0 0419 76 DCX D
0070 03F1 041A 17 MOV A,M
0071 03F2 041B CD0C06 D STAX D ;MOVE DATA
;IF START ADD. IS NOT EQUAL TO END ADD. THEN MOVE AGAIN
0072 03F3 041C 21704 BTUP11 CALL CHLBC ;INCREMENT ENTCT
0073 03F4 0421 D1 BTUP21 JNZ BTUP1 ;RECOVER END ADD. OF DESTINATION DATA
0074 03F5 0422 C3FA03 JMP DSPDE
0075 03F6
0076 03F7 ;***** SAVE PROCESS **
0077 03F8 ;*****
0079 03F9
0080 0425 2AD0B3 FSAV1 LDA ENTCT
0081 0426 A7 ANA A
0082 0427 C23904 JNZ FSAV1 ;IF ENTCT IS GREATER THAN 0 THEN FSAV1
0083 0428 3C INR A
0084 0429 3E STA ENTCT ;INCREMENT ENTCT
0085 0430 3AEC93 LDA DATA
0086 0431 32D703 STA SVDAT ;SAVE FILE NO.
0087 0432 C3E000 JMP ALCLA
0088 0433
0089 0434 0F FSAV11 RRC ;IF ENTCT IS GREATER THAN 1 THEN FSAV11
0090 0435 D2A804 JNC ;A-2
0091 0436 3E MVI A-2
0092 0437 32D8B3 STA ENTCT ;INCREMENT ENTCT
0093 0438 CD7C01 CALL ARSET ;SET START ADD.
0094 0439 445 CD5106 CALL ARCH ;CONVERT (AR) INTO (ADW)
0095 0440 3C3E00 JMP DTCLA
0096 0441
0097 0442 2AEC93 FSAV21 LHL DATA
0098 0443 0444 E8 XCHG ;DE : END ADD.
0099 0444 2AEC93 LHL ADRES ;HL : START ADD.
0100 0445 1D1206 CALL ERMS ;IF START ADD. IS GREATER THAN END ADD. THEN ERROR
0101 0446 05 PUSH D
0102 0447 E2 PUSH H ;SAVE START ADD.

```

9章 モニタプログラムリスト

```

0703 0457 211D05      LXI    H,051DH
0704 045A 22D5E3      SHLD  ADM2      ;GET "S-"
0705 045D 3AD783      LDA    SVDAT
0706 0460 1D7706      CALL  WORDC
0707 0463 22D393      SHLD  ADMA
0708 0466 1D3906      CALL  CD
0709 0469 0C8C05      CALL  SEGCV
0710
0711 046C 0E0D      MVI    C,0
0712 046E CDE706      CALL  READER
0713 0471 BE          CMP    M
0714 0472 3E55      MVI    A,KEYMD
0715 0474 0637      CALL  B,PS4
0716 0476 C0D107      CALL  SRIOT
0717 0479 23      JNZ    H
0718 047A 2B      DCR    M
0719 047B BE          CMP    M
0720
0721 047C 3AD7E3      LDA    SVDAT
0722 047F 0630      MVI    B,PS3
0723 0481 CDD204      CALL  CKSHO
0724 0484 00      NOP
0725 0485 00      NOP
0726 0486 E1      POP    H
0727 0487 D1      POP    D
0728 0488 7E      MOV    A,H
0729 0489 0635      MOV    B,PS3
0730 048B CDD204      CALL  CKSHO
0731 048E 7D      MOV    A,L
0732 048F 0637      MOV    B,PS4
0733 0491 CDD204      CALL  CKSHO
0734 0494 7A      MOV    A,D
0735 0495 0637      MOV    B,PS4
0736 0497 CDD204      CALL  CKSHO
0737 049A 7B      MOV    A,E
0738 049B 0637      MOV    B,PS4
0739 049D CDD204      CALL  CKSHO
0740 04A0 34      JNR    DCR
0741 04A1 35      DCR    M
0742 04A2 79      MOV    A,C
0743 04A3 2F      CMA
0744 04A4 3C      JNR    B
0745 04A5 0635      MVI    B,PS3
0746 04A7 CDD204      CALL  CKSHO
0747
0748 04AA 3AECB3      LDA    DATA
0749 04AD 0635      MVI    B,PS3
0750 04AF 2B      DCR    H
0751 04B0 23      JNZ    H
0752 04B1 7E      MOV    A,H
0753 04B2 CDD204      CALL  CKSHO
0754 04B5 1BF8      IN     PORTA
0755 04B7 0A31      MVI    B,PS1
0756 04B9 CDF05      CALL  COMPA
0757 04BC C2B004      JNZ    FSAV3
0758 04BF 00      NOP
0759 04C0 BE          CMP    M
0760 04C1 79      MOV    A,C
0761 04C2 2F      CMA
0762 04C3 3C      JNR    A
0763 04C4 0630      MVI    B,PS2
0764 04C6 CDD204      CALL  CKSHO
0765
0766 04E9 CD3906      CALL  ED
0767 04CC CDE905      CALL  FLAGC
0768 04CF C37C00      JNP    START
0769
0770 04D2 F5      CKSHD: PUSH  PSH
0771 04D3 81      ADD    C
0772 04D4 4F      MOV    C,A
0773 04D5 F1      POP    PSH
0774 04D6 32FFB3      STA    DIG+7
0775 04D9 C3D107      JNP    SRIOT
0776
0777
0778
0779
0780
0781 04DC 3AECB3      FLOAD: LDA    DATA
0782 04DF F5      PUSH  PSH
0783 04E0 1D7706      CALL  WORDC
0784 04E3 22D393      SHLD  ADMA
0785 04E6 211D05      LXI    H,051DH
0786 04E9 22D393      SHLD  ADM2
0787 04EC CD3906      CALL  CD
0788
0789 04EF CDE605      FILE: CALL  SEGCV
0790 04F2 0650      CALL  C,0
0791 04F4 1DE507      CALL  REGRCH
0792 04F7 CDB307      CALL  SRIIN
0793 04FA FE55      CPI    KEYMD
0794 04FC C2E004      JNZ    FILE
0795
0796 04FF CDE105      CALL  CKSHI
0797 0502 27      MOV    D,A
0798 0503 F1      POP    PSH
0799 0504 BA      CMP    D
0800 0505 CA1305      JZ     FOUND
0801
0802 0508 F5      PUSH  PSH
0803 0509 7A      MOV    A,D
0804 050A CDF706      CALL  WORDC
0805 050D 22CF93      SHLD  DDW4

```

```

0006 0510 C3F04      1      JMP      FILE
0007
0008 0513 AF          FOUND:  XRA      A
0009 0514 32FE93       STA      DIG+6
0010 0517 3E71        MVI      A,71H
0011 0519 32F981       STA      DIG
                                ;DISPLAY FILE FOUND MESSAGE
0012 051C 06105       CALL     CKSMI
0013 051F 67          MOV      H,A
0014 0520 C06105      CALL     CKSMI
0015 0523 AF          MOV      L,A
0016 0524 C06105      CALL     CKSMI
0017 0527 57          MOV      D,A
0018 0528 C06105      CALL     CKSMI
0019 052B 56          MOV      E,A
0020 052C C06105      CALL     CKSMI
0021 052F C29695      JNZ      EREAD
                                ; IF ADDRESS READ ERROR THEN EREAD
0022
0023 0532 E5           PUSH     H
0024 0533 2B          DCI      H
0025 0534 28          INX      H
                                ;SAVE START ADD.
0026 0535 C06105      CALL     CKSMI
                                ;INPUT DATA
0027 0536 77          MOV      M,A
0028 0537 C0FF05      CALL     COMPA
                                ;SET DATA TO MEMORY
0029 053C C23405      JNZ      LOAD1
0030 053F C06105      CALL     CKSMI
0031 0542 61          POP      H
                                ;RECOVER START ADD.
0032 0543 C25A05      JNZ      EREAD
                                ;IF DATA READ ERROR THEN EREAD
0033
0034 0546 22EEF3      SHLD     ADRES
0035 0549 EB          XCHG
                                ;SET START ADD.
0036 054A 22ECB3      SHLD     DATA
0037 054D C04F06      CALL     CH
                                ;SET END ADD.
0038 0550 C0E905      CALL     FLAGC
                                ;CONVERT (AR)-(DR) INTO (ADM)-(DDM)
0039 0553 C37C00      JMP      START
                                ;RESET EACH FLAG
0040
0041 0556 3E0E          EREAD:  MVI      A,0EH
0042 0558 32D6E3       STA      ADWI
                                ;GET "E"
0043 055B CDE905      CALL     FLAGC
                                ;RESET EACH FLAG
0044 055E C3E300      JMP
0045
0046 0561 C06307      CKSMI:  CALL     SRIIN
0047 0564 32F981       STA      DIG+7
                                ;DISPLAY INPUT DATA
0048 0567 47          MOV      B,A
0049 0569 B1          ADD      C
0050 0569 4F          MOV      C,A
0051 056B 78          MOV      A,B
0052 056B C9          RET
0053
0054
0055 *****
0056 *** SEGMENT DATA CONVERSION ***
0057 *****
0058
0059 056C 21D6E3      LTI      H,ADM1
                                ;SET ADD.B DATA DISPLAY REGISTER AND
0060 056D 11F4B3      LXI      D,DIG
                                ;SET SEG BUFF ADD.
0061 0572 01B305      LXI      R,SEGDA
                                ;SET CFG DATA ADD.
0062 0575 3ADF83      LDA      FMODE
                                ;GET MODE FLAG
0063 0578 C47C      ANI      3CH
                                ;SET MASK OF MODE FLAG
0064 057B 07          RLC
0065 057C 07          RLC
0066 057D F4          PUSH     PSW
0067 057E 7E          RGRP:  MOV      A,M
0068 0580 F5          PUSHA
                                ;
0069 0581 2300        MVI      H,0
0070 0583 0A          MOV      L,A
0071 0584 0A          DAD      B
0072 0585 0E          MOV      A,R
                                ;GET SEG DATA
0073 0587 11          MOV      D,SEG DATA TO SEG BUFF
0074 0588 01          POP      H
0075 0589 07          POP     PSW
0076 058A 07          RLC
0077 058B 11          PUSH     PSW
0078 058C 11          CC      PR10D
                                ;IF EACH MODE FLAG IS ON THEN DISPLAY 1 DOT ON (DIG--DIG+3)
0079 058E 2B          DCX      H
0080 0590 01C          INC      E
0081 0592 058E 27D05  JNZ      SEGPR
0082 0594 0591 F1      POP      PSM
                                ;DUMPHY
0083 0595 0591 F1      LDA      FMODE
                                ;GET MODE FLAG
0084 0596 0591 F1      RRC
0085 0597 0596 D2A605  JNC      REGP1
                                ;IF F-MODE IS ON THEN DISPLAY 4 DOTS ON (DIG--DIG+3)
0086 0598 0597 050A8  B4
0087 0599 0598 11F6E3  LXI      D,DIG
0088 059A 0598 C04F05  CALL     PR10D
0089 059B 11          JNZ      INE
0090 059C 059A 05      DCR      B
0091 059D 059A 05      JNZ      SEGMD
0092
0093 059E 059A 3AD6E3  REGP1:  LDA      REG
                                ;GET REGISTER FLAG
0094 05A0 059A 0F      RRC
0095 05A1 059A 10      RRC
0096 05A2 059A 11F6E3  LXI      D,DIG+4
                                ;IF REGISTER FLAG IS ON THEN DISPLAY 1 DOT ON (DIG+4)
0097
0098 05A5 1A          PR10D:  LDAX     D
0099 05A6 05A5 F630      ORI      80H
0100 05A8 05B1 1C      STAX     D
0101 05AB 05B2 C9      RET
0102
0103 05B3 3F065B      SEGDA:  DB      3FH,06H,5BH,4FH,66H,6DH,7DH,27H
0104 05B6 4F666D      DB      7FH,6FH,77H,7CH,39H,5EH,79H,71H
0105 05B9 7D27
0106 05BB 7F6F77
0107 05BD 7C395E
0108 05C1 7971

```

9章 モニタプログラムリスト

```

0905 05C3 761E38          DB      76H,1EH,38H,54H,5CH,73H,67H,50H
05C6 545C73
05C9 6750
0906 05C8 3E1C6E          DB      3EH,1CH,6EH,74H,00H,40H,60H
05CE 740040
05D1 60

0907          I
0908          I *****
0909          I ** SHIF (DR), (DM) **
0910          I *****
0911          I
0912 05D2 2AD0E3          SHIF1: LHL  DMW3
0913 05D5 22D193          SHLD DMW2
0914 05D6 26CFE3          LDA  DMW4
0915 05D8 32D093          STA  DMW3
0916 05DE 2ALEE3          LHL  DATA
0917 05E1 29            DAD  H
0918 05E2 29            DAD  H
0919 05E3 29            DAD  H
0920 05E4 29            DAD  H
0921 05E5 22D093          SHLD DATA
0922 05E6 29            RET

0923          I
0924          I *****
0925          I ** RESET EACH FLAG **
0926          I *****
0927          I
0928 05E9 210000          FLAG1: LRI  H,0
0929 05EC 72D8E3          SHLD ENTCT          ;CLEAR ENTER COUNTER & RESET TRAP FLAG
0930 05F0 2DE93          SHLD FREG          ;RESET REGISTER & MODE FLAG
0931 05F2 3E01          MVI  A,1
0932 05F4 29            RET

0933          I
0934          I *****
0935          I ** SET BIT **
0936          I *****
0937          I
0938 05F5 2C            STMT1: JNR  A          ;SET COUNTER
0939 05F6 4F            MOV  C,A
0940 05F7 3E80          MVI  A,80H
0941 05F9 07            LSHFT: RLC          ;SHIFT LEFT 1 BIT
0942 05FA 00            DCR  C
0943 05FB C2F905          JNZ  LSHFT
0944 05FE 29            RET

0945          I
0946          I
0947          I
0948          I *****
0949          I ** COMPARE (HL) DE, (HL) BC **
0950          I *****
0951          I
0952 05FF 7A            COMP1: MOV  A,D
0953 0600 BC            CMP  H
0954 0601 C2D0B6          JNZ  NTEQU          ;4 STATES DUMMY
0955 0604 00            NOP
0956 0605 7B            MOV  A,E
0957 0606 BD            CMP  L
0958 0607 29            RET
0959 0608 C09000          NTEQU: RET  0          ;9 STATES DUMMY
0960 060B 29            RET
0961          I
0962          I
0963          I
0964          I
0965          I
0966 060C 7B            CHLBC: MOV  A,B
0967 060D BC            CMP  H
0968 060E C0            RNZ  A,C
0969 060F 79            MOV  A,C
0970 0610 BD            CMP  L
0971 0611 29            RET          ;IF HL IS EQUAL TO BC THEN SET ZERO FLAG
0972          I
0973          I
0974          I
0975          I
0976          I
0977 0612 C0FF05          ERRHS1: CALL COMP1
0978 0615 D0            RNC
0979 0616 E1            POP  H
0980 0617 21D0E1          ERR1: LRI  H,1C0EH
0981 0618 22D593          SHLD ADW2
0982 061D 21717H          LXI  H,1717H
0983 0620 22D383          SHLD ADW4
0984 0623 C0E905          CALL FLAGC
0985 0626 C3E300          JMP  DTCLA
0986          I
0987          I
0988          I
0989          I
0990 0629 211C1C          CA1: LXI  H,1C1CH
0991 062C 22D593          SHLD ADW2
0992 062F 22D383          SHLD ADW4          ;PUT OUT ADD.DISPLAY
0993 0632 210900          LXI  H,0
0994 0635 22EEB3          SHLD ADRES          ;SET 0 IN (AR)
0995 0639 29            RET
0996          I
0997 0639 211C1C          CD1: LXI  H,1C1CH
0998 063C 22CFE3          SHLD DMW4
0999 063F AF            XRA  A
1000 0640 32EEB3          STA  DATA          ;SET 0 IN (DR)L
1001 0643 211C1C          CD12: LXI  H,1C1CH
1002 0646 22D183          SHLD DMW2          ;PUT OUT H.DATA DISPLAY

```

```

1003 0A49 AF      XRA      A
1004 0A4A 3EEDR3  STA      DATA+1
1005 0A4D C9      RET
1006
1007
1008
1009
100A
100B
100C
100D
100E
100F
1010
1011 0A4E CD406  CALL     CALL     DMCH
1012
1013 0A51 3AEF83  ANCH:    LDA      ADRES+1
1014 0A54 CD706  CALL     CALL     WORDC
1015 0A57 22DCE3  SHLD     SHLD     ADW2
1016 0A5A 3AE583  LDA      ADRES
1017 0A5D CD706  CALL     CALL     WORDC
1018 0A60 22D933  SHLD     SHLD     ADW4
1019 0A63 C9      RET
1020
1021 0A64 3AEDE3  DMCH:    LDA      DATA+1
1022 0A67 CD706  CALL     CALL     WORDC
1023 0A6A 22D1E3  SHLD     SHLD     DDW2
1024 0A6D 3AEC93  DLWCH:   LDA      DATA
1025 0A70 CD706  CALL     CALL     WORDC
1026 0A73 22CF93  SHLD     SHLD     DDW4
1027 0A76 C9      RET
1028
1029 0A77 47      WORDC:   MOV      B,A
1030 0A78 E50F    ANI      0FH
1031 0A7A 6F      MOV      L,A
1032 0A7B 79      MOV      A,B
1033 0A7C 0F      RRC
1034 0A7D 0F      RRC
1035 0A7E 0F      RRC
1036 0A7F 0F      RRC
1037 0A80 E60F    ANI      0FH
1038 0A82 67      MOV      H,A
1039 0A85 C9      RET
1040
1041
1042
1043
1044
1045 0A86 CD906  KEYIN:   CALL     INPUT
1046 0A89 3ADD93  LDA      FKEY
1047 0A8B A7      ANA      R
1048 0A8E CAB406  JZ       KEYIN
1049 0A91 E878  MOV      A,B
1050 0A94 C9      RET
1051
1052
1053
1054
1055
1056
1057
1058 0A99 CDB06  INPUT:   CALL     KEY
1059 0A9C 3C      INR      A
1060 0A9F CAB706  JZ       NOKEY
1061 0AA2 1A0E  INPRP:   MVI      D,0
1062 0AA5 1E00  MOV      E,D
1063 0AA8 CDBD07  CALL     DELEN
1064 0AAE 15      INR      D
1065 0AB1 C72A06  JNZ      INPRP+2
1066 0AB4 A7      CALL     KEY
1067 0AB7 47      MOV      B,A
1068 0AB9 3E      INR      B
1069 0ABD A7      JZ       NOKEY
1070 0AC0 3ADD33  LDA      FKEY
1071 0AC3 A7      ANA      R
1072 0AC6 C72906  JNZ      INPRP
1073 0AC9 3D      DCR      A
1074 0ACA 3E0E3  SETKF:   STA      FKEY
1075 0ACD 78      MOV      A,B
1076 0AD0 C9      RET
1077 0AD3 0AFF    NOKEY:   MVI      B,0FFH
1078 0AD6 C8206  JMP      SETKF
1079
1080
1081
1082
1083
1084 0AD9 AF      KEY:     XRA      A
1085 0ADB 57      MOV      D,A
1086 0ADE 47      MOV      B,A
1087 0AD9 3EEF    MVI      A,PC4
1088 0AE1 CD406  CALL     SCAN
1089 0AE4 0A03  MVI      B,0
1090 0AE7 3EDF    MVI      A,PC5
1091 0AEA 3E0A06  CALL     SCAN
1092 0AED 0A10  MVI      B,10H
1093 0AF0 3EBF    MVI      A,PC6
1094 0AF3 CD406  CALL     SCAN
1095 0AF6 3D      DCR      A
1096 0AF9 C9      RET
1097 0ADA D2FA  SCAN:   OUT      PORTC
1098 0ADB DBF9  IN       PORTA
1099 0AE6 2F      CMA
1100 0AD7 A7      ANA      A
1101 0ADA C9      RZ
1102 0AD9 E1      POP      H
1103 0ADE 0F      KEY:     RRC

```

9章 モニタプログラムリスト

```

1104 0ADD DAE06          JC      KEY2
1105 06E0 14             INR      D
1106 0AE1 C3DC06         JMP      KEY1
1107 0AE4 7A             MOV      A,D
1108 0AE5 B0             ORA      B
1109 0AE6 C9             RET
1110
1111 I
1112 I*****
1113 I** OUTPUT READER FOR 5SEC **
1114 I*****
1115 06E7 C5             I
1116 0AE8 E5             REDER:  PUSH  B          ;SAVE BC
1117 0AE9 217017         PUSH  H          ;SAVE HL
1118                               LXI   H,5E15      ;SET LOOP COUNTER
1119 I
1120 06EC CD3B07         REDR1:  CALL  ONEOT      ;OUTPUT '1'
1121 0AEF 46             MOV      B,M          ;7 STATES DUMMY
1122 06F0 061C         MOV      B,FRA        ;SET DELAY COUNTER
1123 06F2 2D             DCR      L
1124 06F3 C2EC06         JNZ      REDR1
1125 06F6 DBF8         I
1126 06F9 061A         IN      PORTA        ;10 STATES DUMMY
1127 06FA 25             MVI      B,PRB        ;SET DELAY COUNTER
1128 06FB C2EC06         DCR      H
1129 06FC 061A         JNZ      REDR1
1130 06FE E1             POP      H          ;RECOVER HL
1131 06FF C1             POP      B          ;RECOVER BC
1132 0700 C9             RET
1133 I
1134 I*****
1135 I** OUTPUT ONE BYTE **
1136 I*****
1137 I
1138 0701 C5             SROT1:  PUSH  B          ;SAVE BC
1139 0702 D5             PUSH  D          ;SAVE DE
1140 0703 E5             PUSH  H          ;SAVE HL
1141 I
1142 0704 4F             MOV      C,A          ;SAVE 1 BYTE DATA
1143 0705 CDAA07         CALL  ZEROT      ;WRITE START BIT (0)
1144 0706 013E19         LXI   D,P10        ;SET DELAY COUNTER
1145 070B 7E             MOV      A,M          ;7 STATES DUMMY
1146 070C 2E0B         MOV      L,B          ;SET LOOP COUNTER
1147 I
1148 070E 79             SROT1:  MOV      A,C          ;RECOVER 1 BYTE DATA
1149 070F 1F             RAR
1150 0710 4F             MOV      C,A          ;SAVE 1 BYTE DATA
1151 0711 DA1E07         JC      SROT2      ;IF DATA IS EQUAL TO '1' THEN POP
1152 I
1153 0714 43             MOV      B,E          ;SET DELAY COUNTER
1154 0715 DBF8         IN      PORTA        ;10 STATES DUMMY
1155 0717 7E             MOV      A,M          ;7 STATES DUMMY
1156 0718 CDAA07         CALL  ZEROT      ;WRITE 1 BIT (0)
1157 071B C32B07         JMP      SROT3
1158 I
1159 071E 42             SROT2:  MOV      B,D          ;SET DELAY COUNTER
1160 071F DBF8         IN      PORTA        ;10 STATES DUMMY
1161 0721 DBF8         IN      PORTA        ;10 STATES DUMMY
1162 0723 CD3B07         CALL  ONEOT      ;WRITE 1 BIT (1)
1163 0726 DBF8         IN      PORTA        ;10 STATES DUMMY
1164 I
1165 072B 2D             SROT3:  DCR      L
1166 0729 C20E07         JNZ      SROT1      ;IF END OF 8 BITS THEN CONTINUE
1167 I
1168 072C 061C         MOV      B,POA        ;SET DELAY COUNTER
1169 072E CD3B07         CALL  ONEOT      ;WRITE END BIT (1)
1170 0731 7E             MOV      A,M          ;7 STATES DUMMY
1171 0732 061D         MVI      B,POB        ;SET DELAY COUNTER
1172 0734 CD3B07         CALL  ONEOT      ;WRITE END BIT (1)
1173 I
1174 0737 E1             POP      H          ;RECOVER HL
1175 0738 D1             POP      D          ;RECOVER DE
1176 0739 C1             POP      B          ;RECOVER BC
1177 073A C9             RET
1178 I
1179 I*****
1180 I** OUTPUT ONE BIT **
1181 I*****
1182 I
1183 073B 3E0B         ONEOT:  MVI      A,HIGH        ;SET WAVE TO HIGH
1184 073D CD5907         CALL  WVC01      ;CHANGE WAVE FROM HIGH TO LOW
1185 0740 CD5707         CALL  WVC02      ;CHANGE WAVE FROM LOW TO HIGH
1186 0743 CD5707         CALL  WVC03      ;CHANGE WAVE FROM HIGH TO LOW
1187 0746 CD5707         CALL  WVC04      ;CHANGE WAVE FROM LOW TO HIGH
1188 0749 C9             RET
1189 I
1190 074A 3E0B         ZEROT:  MVI      A,HIGH        ;SET WAVE TO HIGH
1191 074C CD5907         CALL  WVC01      ;CHANGE WAVE FROM HIGH TO LOW
1192 074F 00             NOP
1193 0750 00             NOP
1194 0751 0644         MVI      B,POD        ;SET DELAY COUNTER
1195 0753 CD5907         CALL  WVC01      ;CHANGE WAVE FROM LOW TO HIGH
1196 0756 C9             RET
1197 I
1198 0757 0620         WVC01:  MVI      B,PO1        ;SET DELAY COUNTER
1199 0759 05             WVC01:  DCR      B
1200 075A C25907         JNZ      WVC01      ;DELAY HALF CYCLE
1201 075D 03             INX      B
1202 075E 0B             DCR      B
1203 075F 00             MVI      B,PO1        ;16 STATES DUMMY
1204 0760 03             INX      B
1205 0761 0B             DCR      B
1206 0762 00             MVI      B,PO1        ;14 STATES DUMMY
1207 0763 03             INX      B
1208 0764 0B             DCR      B
1209 0765 00             MVI      B,PO1        ;16 STATES DUMMY
1210 0766 03             INX      B
1211 0767 0B             DCR      B
1212 0768 00             MVI      B,PO1        ;14 STATES DUMMY
1213 0769 03             INX      B
1214 076A 0B             DCR      B
1215 076B 00             MVI      B,PO1        ;16 STATES DUMMY
1216 076C 03             INX      B
1217 076D 0B             DCR      B
1218 076E 00             MVI      B,PO1        ;14 STATES DUMMY
1219 076F 03             INX      B
1220 0770 0B             DCR      B
1221 0771 00             MVI      B,PO1        ;16 STATES DUMMY
1222 0772 03             INX      B
1223 0773 0B             DCR      B
1224 0774 00             MVI      B,PO1        ;14 STATES DUMMY
1225 0775 03             INX      B
1226 0776 0B             DCR      B
1227 0777 00             MVI      B,PO1        ;16 STATES DUMMY
1228 0778 03             INX      B
1229 0779 0B             DCR      B
1230 077A 00             MVI      B,PO1        ;14 STATES DUMMY
1231 077B 03             INX      B
1232 077C 0B             DCR      B
1233 077D 00             MVI      B,PO1        ;16 STATES DUMMY
1234 077E 03             INX      B
1235 077F 0B             DCR      B
1236 0780 00             MVI      B,PO1        ;14 STATES DUMMY
1237 0781 03             INX      B
1238 0782 0B             DCR      B
1239 0783 00             MVI      B,PO1        ;16 STATES DUMMY
1240 0784 03             INX      B
1241 0785 0B             DCR      B
1242 0786 00             MVI      B,PO1        ;14 STATES DUMMY
1243 0787 03             INX      B
1244 0788 0B             DCR      B
1245 0789 00             MVI      B,PO1        ;16 STATES DUMMY
1246 078A 03             INX      B
1247 078B 0B             DCR      B
1248 078C 00             MVI      B,PO1        ;14 STATES DUMMY
1249 078D 03             INX      B
1250 078E 0B             DCR      B
1251 078F 00             MVI      B,PO1        ;16 STATES DUMMY
1252 0790 03             INX      B
1253 0791 0B             DCR      B
1254 0792 00             MVI      B,PO1        ;14 STATES DUMMY
1255 0793 03             INX      B
1256 0794 0B             DCR      B
1257 0795 00             MVI      B,PO1        ;16 STATES DUMMY
1258 0796 03             INX      B
1259 0797 0B             DCR      B
1260 0798 00             MVI      B,PO1        ;14 STATES DUMMY
1261 0799 03             INX      B
1262 079A 0B             DCR      B
1263 079B 00             MVI      B,PO1        ;16 STATES DUMMY
1264 079C 03             INX      B
1265 079D 0B             DCR      B
1266 079E 00             MVI      B,PO1        ;14 STATES DUMMY
1267 079F 03             INX      B
1268 07A0 0B             DCR      B
1269 07A1 00             MVI      B,PO1        ;16 STATES DUMMY
1270 07A2 03             INX      B
1271 07A3 0B             DCR      B
1272 07A4 00             MVI      B,PO1        ;14 STATES DUMMY
1273 07A5 03             INX      B
1274 07A6 0B             DCR      B
1275 07A7 00             MVI      B,PO1        ;16 STATES DUMMY
1276 07A8 03             INX      B
1277 07A9 0B             DCR      B
1278 07AA 00             MVI      B,PO1        ;14 STATES DUMMY
1279 07AB 03             INX      B
1280 07AC 0B             DCR      B
1281 07AD 00             MVI      B,PO1        ;16 STATES DUMMY
1282 07AE 03             INX      B
1283 07AF 0B             DCR      B
1284 07B0 00             MVI      B,PO1        ;14 STATES DUMMY
1285 07B1 03             INX      B
1286 07B2 0B             DCR      B
1287 07B3 00             MVI      B,PO1        ;16 STATES DUMMY
1288 07B4 03             INX      B
1289 07B5 0B             DCR      B
1290 07B6 00             MVI      B,PO1        ;14 STATES DUMMY
1291 07B7 03             INX      B
1292 07B8 0B             DCR      B
1293 07B9 00             MVI      B,PO1        ;16 STATES DUMMY
1294 07BA 03             INX      B
1295 07BB 0B             DCR      B
1296 07BC 00             MVI      B,PO1        ;14 STATES DUMMY
1297 07BD 03             INX      B
1298 07BE 0B             DCR      B
1299 07BF 00             MVI      B,PO1        ;16 STATES DUMMY
1300 07C0 03             INX      B
1301 07C1 0B             DCR      B
1302 07C2 00             MVI      B,PO1        ;14 STATES DUMMY
1303 07C3 03             INX      B
1304 07C4 0B             DCR      B
1305 07C5 00             MVI      B,PO1        ;16 STATES DUMMY
1306 07C6 03             INX      B
1307 07C7 0B             DCR      B
1308 07C8 00             MVI      B,PO1        ;14 STATES DUMMY
1309 07C9 03             INX      B
1310 07CA 0B             DCR      B
1311 07CB 00             MVI      B,PO1        ;16 STATES DUMMY
1312 07CC 03             INX      B
1313 07CD 0B             DCR      B
1314 07CE 00             MVI      B,PO1        ;14 STATES DUMMY
1315 07CF 03             INX      B
1316 07D0 0B             DCR      B
1317 07D1 00             MVI      B,PO1        ;16 STATES DUMMY
1318 07D2 03             INX      B
1319 07D3 0B             DCR      B
1320 07D4 00             MVI      B,PO1        ;14 STATES DUMMY
1321 07D5 03             INX      B
1322 07D6 0B             DCR      B
1323 07D7 00             MVI      B,PO1        ;16 STATES DUMMY
1324 07D8 03             INX      B
1325 07D9 0B             DCR      B
1326 07DA 00             MVI      B,PO1        ;14 STATES DUMMY
1327 07DB 03             INX      B
1328 07DC 0B             DCR      B
1329 07DD 00             MVI      B,PO1        ;16 STATES DUMMY
1330 07DE 03             INX      B
1331 07DF 0B             DCR      B
1332 07E0 00             MVI      B,PO1        ;14 STATES DUMMY
1333 07E1 03             INX      B
1334 07E2 0B             DCR      B
1335 07E3 00             MVI      B,PO1        ;16 STATES DUMMY
1336 07E4 03             INX      B
1337 07E5 0B             DCR      B
1338 07E6 00             MVI      B,PO1        ;14 STATES DUMMY
1339 07E7 03             INX      B
1340 07E8 0B             DCR      B
1341 07E9 00             MVI      B,PO1        ;16 STATES DUMMY
1342 07EA 03             INX      B
1343 07EB 0B             DCR      B
1344 07EC 00             MVI      B,PO1        ;14 STATES DUMMY
1345 07ED 03             INX      B
1346 07EE 0B             DCR      B
1347 07EF 00             MVI      B,PO1        ;16 STATES DUMMY
1348 07F0 03             INX      B
1349 07F1 0B             DCR      B
1350 07F2 00             MVI      B,PO1        ;14 STATES DUMMY
1351 07F3 03             INX      B
1352 07F4 0B             DCR      B
1353 07F5 00             MVI      B,PO1        ;16 STATES DUMMY
1354 07F6 03             INX      B
1355 07F7 0B             DCR      B
1356 07F8 00             MVI      B,PO1        ;14 STATES DUMMY
1357 07F9 03             INX      B
1358 07FA 0B             DCR      B
1359 07FB 00             MVI      B,PO1        ;16 STATES DUMMY
1360 07FC 03             INX      B
1361 07FD 0B             DCR      B
1362 07FE 00             MVI      B,PO1        ;14 STATES DUMMY
1363 07FF 03             INX      B
1364 0800 0B             DCR      B
1365 0801 00             MVI      B,PO1        ;16 STATES DUMMY
1366 0802 03             INX      B
1367 0803 0B             DCR      B
1368 0804 00             MVI      B,PO1        ;14 STATES DUMMY
1369 0805 03             INX      B
1370 0806 0B             DCR      B
1371 0807 00             MVI      B,PO1        ;16 STATES DUMMY
1372 0808 03             INX      B
1373 0809 0B             DCR      B
1374 080A 00             MVI      B,PO1        ;14 STATES DUMMY
1375 080B 03             INX      B
1376 080C 0B             DCR      B
1377 080D 00             MVI      B,PO1        ;16 STATES DUMMY
1378 080E 03             INX      B
1379 080F 0B             DCR      B
1380 0810 00             MVI      B,PO1        ;14 STATES DUMMY
1381 0811 03             INX      B
1382 0812 0B             DCR      B
1383 0813 00             MVI      B,PO1        ;16 STATES DUMMY
1384 0814 03             INX      B
1385 0815 0B             DCR      B
1386 0816 00             MVI      B,PO1        ;14 STATES DUMMY
1387 0817 03             INX      B
1388 0818 0B             DCR      B
1389 0819 00             MVI      B,PO1        ;16 STATES DUMMY
1390 081A 03             INX      B
1391 081B 0B             DCR      B
1392 081C 00             MVI      B,PO1        ;14 STATES DUMMY
1393 081D 03             INX      B
1394 081E 0B             DCR      B
1395 081F 00             MVI      B,PO1        ;16 STATES DUMMY
1396 0820 03             INX      B
1397 0821 0B             DCR      B
1398 0822 00             MVI      B,PO1        ;14 STATES DUMMY
1399 0823 03             INX      B
1400 0824 0B             DCR      B
1401 0825 00             MVI      B,PO1        ;16 STATES DUMMY
1402 0826 03             INX      B
1403 0827 0B             DCR      B
1404 0828 00             MVI      B,PO1        ;14 STATES DUMMY
1405 0829 03             INX      B
1406 082A 0B             DCR      B
1407 082B 00             MVI      B,PO1        ;16 STATES DUMMY
1408 082C 03             INX      B
1409 082D 0B             DCR      B
1410 082E 00             MVI      B,PO1        ;14 STATES DUMMY
1411 082F 03             INX      B
1412 0830 0B             DCR      B
1413 0831 00             MVI      B,PO1        ;16 STATES DUMMY
1414 0832 03             INX      B
1415 0833 0B             DCR      B
1416 0834 00             MVI      B,PO1        ;14 STATES DUMMY
1417 0835 03             INX      B
1418 0836 0B             DCR      B
1419 0837 00             MVI      B,PO1        ;16 STATES DUMMY
1420 0838 03             INX      B
1421 0839 0B             DCR      B
1422 083A 00             MVI      B,PO1        ;14 STATES DUMMY
1423 083B 03             INX      B
1424 083C 0B             DCR      B
1425 083D 00             MVI      B,PO1        ;16 STATES DUMMY
1426 083E 03             INX      B
1427 083F 0B             DCR      B
1428 0840 00             MVI      B,PO1        ;14 STATES DUMMY
1429 0841 03             INX      B
1430 0842 0B             DCR      B
1431 0843 00             MVI      B,PO1        ;16 STATES DUMMY
1432 0844 03             INX      B
1433 0845 0B             DCR      B
1434 0846 00             MVI      B,PO1        ;14 STATES DUMMY
1435 0847 03             INX      B
1436 0848 0B             DCR      B
1437 0849 00             MVI      B,PO1        ;16 STATES DUMMY
1438 084A 03             INX      B
1439 084B 0B             DCR      B
1440 084C 00             MVI      B,PO1        ;14 STATES DUMMY
1441 084D 03             INX      B
1442 084E 0B             DCR      B
1443 084F 00             MVI      B,PO1        ;16 STATES DUMMY
1444 0850 03             INX      B
1445 0851 0B             DCR      B
1446 0852 00             MVI      B,PO1        ;14 STATES DUMMY
1447 0853 03             INX      B
1448 0854 0B             DCR      B
1449 0855 00             MVI      B,PO1        ;16 STATES DUMMY
1450 0856 03             INX      B
1451 0857 0B             DCR      B
1452 0858 00             MVI      B,PO1        ;14 STATES DUMMY
1453 0859 03             INX      B
1454 085A 0B             DCR      B
1455 085B 00             MVI      B,PO1        ;16 STATES DUMMY
1456 085C 03             INX      B
1457 085D 0B             DCR      B
1458 085E 00             MVI      B,PO1        ;14 STATES DUMMY
1459 085F 03             INX      B
1460 0860 0B             DCR      B
1461 0861 00             MVI      B,PO1        ;16 STATES DUMMY
1462 0862 03             INX      B
1463 0863 0B             DCR      B
1464 0864 00             MVI      B,PO1        ;14 STATES DUMMY
1465 0865 03             INX      B
1466 0866 0B             DCR      B
1467 0867 00             MVI      B,PO1        ;16 STATES DUMMY
1468 0868 03             INX      B
1469 0869 0B             DCR      B
1470 086A 00             MVI      B,PO1        ;14 STATES DUMMY
1471 086B 03             INX      B
1472 086C 0B             DCR      B
1473 086D 00             MVI      B,PO1        ;16 STATES DUMMY
1474 086E 03             INX      B
1475 086F 0B             DCR      B
1476 0870 00             MVI      B,PO1        ;14 STATES DUMMY
1477 0871 03             INX      B
1478 0872 0B             DCR      B
1479 0873 00             MVI      B,PO1        ;16 STATES DUMMY
1480 0874 03             INX      B
1481 0875 0B             DCR      B
1482 0876 00             MVI      B,PO1        ;14 STATES DUMMY
1483 0877 03             INX      B
1484 0878 0B             DCR      B
1485 0879 00             MVI      B,PO1        ;16 STATES DUMMY
1486 087A 03             INX      B
1487 087B 0B             DCR      B
1488 087C 00             MVI      B,PO1        ;14 STATES DUMMY
1489 087D 03             INX      B
1490 087E 0B             DCR      B
1491 087F 00             MVI      B,PO1        ;16 STATES DUMMY
1492 0880 03             INX      B
1493 0881 0B             DCR      B
1494 0882 00             MVI      B,PO1        ;14 STATES DUMMY
1495 0883 03             INX      B
1496 0884 0B             DCR      B
1497 0885 00             MVI      B,PO1        ;16 STATES DUMMY
1498 0886 03             INX      B
1499 0887 0B             DCR      B
1500 0888 00             MVI      B,PO1        ;14 STATES DUMMY
1501 0889 03             INX      B
1502 088A 0B             DCR      B
1503 088B 00             MVI      B,PO1        ;16 STATES DUMMY
1504 088C 03             INX      B
1505 088D 0B             DCR      B
1506 088E 00             MVI      B,PO1        ;14 STATES DUMMY
1507 088F 03             INX      B
1508 0890 0B             DCR      B
1509 0891 00             MVI      B,PO1        ;16 STATES DUMMY
1510 0892 03             INX      B
1511 0893 0B             DCR      B
1512 0894 00             MVI      B,PO1        ;14 STATES DUMMY
1513 0895 03             INX      B
1514 0896 0B             DCR      B
1515 0897 00             MVI      B,PO1        ;16 STATES DUMMY
1516 0898 03             INX      B
1517 0899 0B             DCR      B
1518 089A 00             MVI      B,PO1        ;14 STATES DUMMY
1519 089B 03             INX      B
1520 089C 0B             DCR      B
1521 089D 00             MVI      B,PO1        ;16 STATES DUMMY
1522 089E 03             INX      B
1523 089F 0B             DCR      B
1524 08A0 00             MVI      B,PO1        ;14 STATES DUMMY
1525 08A1 03             INX      B
1526 08A2 0B             DCR      B
1527 08A3 00             MVI      B,PO1        ;16 STATES DUMMY
1528 08A4 03             INX      B
1529 08A5 0B             DCR      B
1530 08A6 00             MVI      B,PO1        ;14 STATES DUMMY
1531 08A7 03             INX      B
1532 08A8 0B             DCR      B
1533 08A9 00             MVI      B,PO1        ;16 STATES DUMMY
1534 08AA 03             INX      B
1535 08AB 0B             DCR      B
1536 08AC 00             MVI      B,PO1        ;14 STATES DUMMY
1537 08AD 03             INX      B
1538 08AE 0B             DCR      B
1539 08AF 00             MVI      B,PO1        ;16 STATES DUMMY
1540 08B0 03             INX      B
1541 08B1 0B             DCR      B
1542 08B2 00             MVI      B,PO1        ;14 STATES DUMMY
1543 08B3 03             INX      B
1544 08B4 0B             DCR      B
1545 08B5 00             MVI      B,PO1        ;16 STATES DUMMY
1546 08B6 03             INX      B
1547 08B7 0B             DCR      B
1548 08B8 00             MVI      B,PO1        ;14 STATES DUMMY
1549 08B9 03             INX      B
1550 08BA 0B             DCR      B
1551 08BB 00             MVI      B,PO1        ;16 STATES DUMMY
1552 08BC 03             INX      B
1553 08BD 0B             DCR      B
1554 08BE 00             MVI      B,PO1        ;14 STATES DUMMY
1555 08BF 03             INX      B
1556 08C0 0B             DCR      B
1557 08C1 00             MVI      B,PO1        ;16 STATES DUMMY
1558 08C2 03             INX      B
1559 08C3 0B             DCR      B
1560 08C4 00             MVI      B,PO1        ;14 STATES DUMMY
1561 08C5 03             INX      B
1562 08C6 0B             DCR      B
1563 08C7 00             MVI      B,PO1        ;16 STATES DUMMY
1564 08C8 03             INX      B
1565 08C9 0B             DCR      B
1566 08CA 00             MVI      B,PO1        ;14 STATES DUMMY
1567 08CB 03             INX      B
1568 08CC 0B             DCR      B
1569 08CD 00             MVI      B,PO1        ;16 STATES DUMMY
1570 08CE 03             INX      B
1571 08CF 0B             DCR      B
1572 08D0 00             MVI      B,PO1        ;14 STATES DUMMY
1573 08D1 03             INX      B
1574 08D2 0B             DCR      B
1575 08D3 00             MVI      B,PO1        ;16 STATES DUMMY
1576 08D4 03             INX      B
1577 08D5 0B             DCR      B
1578 08D6 00             MVI      B,PO1        ;14 STATES DUMMY
1579 08D7 03             INX      B
1580 08D8 0B             DCR      B
1581 08D9 00             MVI      B,PO1        ;16 STATES DUMMY
1582 08DA 03             INX      B
1583 08DB 0B             DCR      B
1584 08DC 00             MVI      B,PO1        ;14 STATES DUMMY
1585 08DD 03             INX      B
1586 08DE 0B             DCR      B
1587 08DF 00             MVI      B,PO1        ;16 STATES DUMMY
1588 08E0 03             INX      B
1589 08E1 0B             DCR      B
1590 08E2 00             MVI      B,PO1        ;14 STATES DUMMY
1591 08E3 03             INX      B
1592 08E4 0B             DCR      B
1593 08E5 00             MVI      B,PO1        ;16 STATES DUMMY
1594 08E6 03             INX      B
1595 08E7 0B             DCR      B
1596 08E8 00             MVI      B,PO1        ;14 STATES DUMMY
1597 08E9 03             INX      B
1598 08EA 0B             DCR      B
1599 08EB 00             MVI      B,PO1        ;16 STATES DUMMY
1600 08EC 03             INX      B
1601 08ED 0B             DCR      B
1602 08EE 00             MVI      B,PO1        ;14 STATES DUMMY
1603 08EF 03             INX      B
1604 08F0 0B             DCR      B
1605 08F1 00             MVI      B,PO1        ;16 STATES DUMMY
1606 08F2 03             INX      B
1607 08F3 0B             DCR      B
1608 08F4 00             MVI      B,PO1        ;14 STATES DUMMY
1609 08F5 03             INX      B
1610 08F6 0B             DCR      B
1611 08F7 00             MVI      B,PO1        ;16 STATES DUMMY
1612 08F8 03             INX      B
1613 08F9 0B             DCR      B
1614 08FA 00             MVI      B,PO1        ;14 STATES DUMMY
1615 08FB 03             INX      B
1616 08FC 0B             DCR      B
1617 08FD 00             MVI      B,PO1        ;16 STATES DUMMY
1618 08FE 03             INX      B
1619 08FF 0B             DCR      B
1620 0900 00             MVI      B,PO1        ;14 STATES DUMMY
1621 0901 03             INX      B
1622 0902 0B             DCR      B
1623 0903 00             MVI      B,PO1        ;16 STATES DUMMY
1624 0904 03             INX      B
1625 0905 0B             DCR      B
1626 0906 00             MVI      B,PO1        ;14 STATES DUMMY
1627 0907 03             INX      B
1628 0908 0B             DCR      B
1629 0909 00             MVI      B,PO1        ;16 STATES DUMMY
1630 090A 03             INX      B
1631 090B 0B             DCR      B
1632 090C 00             MVI      B,PO1        ;14 STATES DUMMY
1633 090D 03             INX      B
1634 090E 0B             DCR      B
1635 090F 00             MVI      B,PO1        ;16 STATES DUMMY
1636 0910 03             INX      B
1637 0911 0B             DCR      B
1638 0912 00             MVI      B,PO1        ;14 STATES DUMMY
1639 0913 03             INX      B
1640 0914 0B             DCR      B
1641 0915 00             MVI      B,PO1        ;16 STATES DUMMY
1642 0916 03             INX      B
1643 0917 0B             DCR      B
1644 0918 00             MVI      B,PO1        ;14 STATES DUMMY
1645 0919 03             INX      B
1646 091A 0B             DCR      B
1647 091B 00             MVI      B,PO1        ;16 STATES DUMMY
1648 091C 03             INX      B
1649 091D 0B             DCR      B
1650 091E 00             MVI      B,PO1        ;14 STATES DUMMY
1651 091F 03             INX      B
1652 0920 0B             DCR      B
1653 0921 00             MVI      B,PO1        ;16 STATES DUMMY
1654 0922 03             INX      B
1655 0923 0B             DCR      B
1656 0924 00             MVI      B,PO1        ;14 STATES DUMMY
1657 0925 03             INX      B
1658 0926 0B             DCR      B
1659 0927 00             MVI      B,PO1        ;16 STATES DUMMY
1660 0928 03             INX      B
1661 0929 0B             DCR      B
1662 092A 00             MVI      B,PO1        ;14 STATES DUMMY
1663 092B 03             INX      B
1664 092C 0B             DCR      B
1665 092D 00             MVI      B,PO1        ;16 STATES DUMMY
1666 092E 03             INX      B
1667 092F 0B             DCR      B
1668 0930 00             MVI      B,PO1        ;14 STATES DUMMY
1669 0931 03             INX      B
1670 0932 0B             DCR      B
1671 0933 00             MVI      B,PO1        ;16 STATES DUMMY
1672 0934 03             INX      B
1673 0935 0B             DCR      B
1674 0936 00             MVI      B,PO1        ;14 STATES DUMMY
1675 0937 03             INX      B
1676 0938 0B             DCR      B
1677 0939 00             MVI      B,PO1        ;16 STATES DUMMY
1678 093A 03             INX      B
1679 093B 0B             DCR      B
1680 093C 00             MVI      B,PO1        ;14 STATES DUMMY
1681 093D 03             INX      B
1682 093E 0B             DCR      B
1683 093F 00             MVI      B,PO1        ;16 STATES DUMMY
1684 0940 03             INX      B
1685 0941 0B             DCR      B
1686 0942 00             MVI      B,PO1        ;14 STATES DUMMY
1687 0943 03             INX      B
1688 0944 0B             DCR      B
1689 0945 00             MVI      B,PO1        ;16 STATES DUMMY
1690 0946 03             INX      B
1691 0947 0B             DCR      B
169
```

```

1204 0760 17      RAL          ;
1205 0761 3F      CHC          ;CHANGE WAVE DATA
1206 0762 1F      RAR          ;
1207 0763 30      SIN          ;OUTPUT SERIAL DATA
1208 0764 C9      RET
1209
1210
1211 *****
1212 *****
1213 *****
1214 0765 C5      RDSCH: PUSH    B          ;SAVE BC
1215 0766 D5      PUSH    D          ;SAVE DE
1216
1217 0767 20      ;
1218 0768 47      MOV     B,A          ;SERIAL INPUT
1219 0769 1600    RDSCH: MVI     D,0      ;SAVE INPUT DATA
1220 076B 1E00    RDSCH: MVI     E,0      ;CLEAR 256 COUNTER
1221 076D CDC607  CALL     SERCH        ;CLEAR DELAY COUNTER
1222 0770 3E1F    MVI     A,BIMAX      ;SEARCH WAVE CHANGE
1223 0772 BB      CMP     E          ;SET DELAY COUNTER TO MAX. THRESHOLD OF DATA '1'
1224 0773 D46907 JC      RDSCH1        ;IF NOT READER THEN SEARCH READER AGAIN
1225 0776 3E09    MVI     A,BIMIN      ;SET DELAY COUNTER TO MIN. THRESHOLD OF DATA '1'
1226 0778 BB      CMP     E          ;IF NOT READER THEN SEARCH READER AGAIN
1227 0779 D26907 JNC     RDSCH1
1228 077C 14      INR     D          ;
1229 077D C26B07 JNZ     RDSCH2        ;IF FREQUENCY OF 256 WAVES IS EQUAL TO 2400HZ THEN CONTINUE
1230
1231 0780 D1      POP     D          ;RECOVER DE
1232 0781 C1      POP     B          ;RECOVER BC
1233 0782 C9      RET
1234
1235 *****
1236 *****
1237 *****
1238 *****
1239 0783 C5      SRIIN: PUSH    B          ;SAVE BC
1240 0784 D5      PUSH    D          ;SAVE DE
1241 0785 E5      PUSH    H          ;SAVE HL
1242
1243 0786 20      ;
1244 0787 47      MOV     B,A          ;SERIAL INPUT
1245 0788 1E00    SRII1: MVI     E,0      ;SAVE INPUT DATA
1246 078A CDC607  CALL     SERCH        ;CLEAR DELAY COUNTER
1247 078D 3E20    MVI     A,BOFWD      ;SEARCH WAVE CHANGE
1248 078F BB      CMP     E          ;SET DELAY COUNTER TO THRESHOLD OF DATA '0'
1249 0790 D28B07 JNC     SRII1        ;IF DATA IS EQUAL TO 1 THEN SEARCH START BIT AGAIN
1250
1251 0793 3E4A      MVI     A,BORCK      ;SET DELAY COUNTER
1252 0795 CDBF07  CALL     DELE1        ;DELAY
1253 0798 2E0B      MVI     L,B        ;SET BITS LOOP COUNTER
1254 079A CBA007  CALL     ONEIN        ;INPUT ONE BIT DATA
1255 079D 7C      MOV     A,H          ;LOAD ONE BYTE DATA
1256 079E 1F      RAR          ;
1257 079F 67      MOV     H,A          ;STORE ONE BYTE DATA
1258 07A0 CDBD07  CALL     DELEN        ;DELAY
1259 07A3 2D      DCR     L          ;
1260 07A4 C29A07 JNZ     SRII2        ;IF END OF 8 BITS THEN CONTINUE
1261 07A7 7C      MOV     A,H          ;SET ONE BYTE DATA TO A
1262
1263 07AB E1      POP     H          ;RECOVER HL
1264 07AD D1      POP     D          ;RECOVER DE
1265 07AE C1      POP     B          ;RECOVER BC
1266 07AB C9      RET
1267
1268 *****
1269 *****
1270 *****
1271 *****
1272 07AC CDC607  ONEIN: CALL     SERCH        ;SEARCH WAVE CHANGE
1273 07AF 3E35      MVI     A,C50TH      ;SET DELAY COUNTER
1274 07B1 1F00      MVI     E,0          ;CLEAR DELAY COUNTER
1275 07B3 CDBF07  CALL     DELE1        ;DELAY FOR 5/8 CYCLE
1276 07B6 CDC607  CALL     SERCH        ;SEARCH WAVE CHANGE
1277 07B9 3E37      MVI     A,C50TH+2    ;LOAD DATA THRESHOLD
1278 07BB BB      CMP     E          ;
1279 07BC C9      RET          ;IF ONE BIT DATA IS EQUAL TO '1' THEN SET CARRY
1280
1281 *****
1282 *****
1283 *****
1284 *****
1285 *****
1286 07BD 3E47      DELEN: MVI     A,DELAY    ;SET DELAY COUNTER
1287 07BF 1C      DELE1: INR     E          ;
1288 07C0 0D      NOP          ;
1289 07C1 BB      CMP     E          ;4 STATES DUMMY
1290 07C2 D2BF07  JNC     DELE1
1291 07C5 C9      RET
1292
1293 *****
1294 *****
1295 *****
1296 *****
1297 07CA 1C      SERCH: INR     E          ;
1298 07CB 20      RIM          ;SERIAL INPUT
1299 07CC AB      XRA          ;
1300 07CD 1C07      JZ      SERCH        ;WAVE NOT CHANGE THEN SEARCH AGAIN
1301 07CC AB      XRA          ;
1302 07CD 47      MOV     B,A          ;SAVE INPUT DATA

```

9章 モニタプログラムリスト

```

1303 07CE C9          RET
1304
1305
1306
1307
1308
1309 07CF 21EF83      RGDSPI: LXI    H,ADRES+1
1310 07D2 11F483      LXI    D,DISP
1311 07D5 0604        MVI    B,B-4
1312 07D7 7E         RGDSI: MOV    A,M
1313 07D9 12          STAX    D
1314 07DB 2B          DCR     H
1315 07DA 13          INX     D
1316 07DB 05          DCR     B
1317 07DC C2D707     JNZ     RGDSI
1318
1319
1320
1321
1322
1323 07E3 07E4 0C04    SEGC0: MVI    B,B-4
1324 07E1 21F483      LXI    H,DISP
1325 07E4 11F483      LXI    D,ADW1
1326 07E7 7E         SEGC1: MOV    A,M
1327 07E6 C1F707     CALL   SEGC2
1328 07E8 7E         MOV    A,M
1329 07E9 CDF707     CALL   SEGC3
1330 07E7 23          INX     H
1331 07F0 05          DCR     B
1332 07F1 C2E707     JNZ     SEGC1
1333 07F4 C8C005     JMP     SEGC0
1334 07F7 0F         SEGC2: RRC
1335 07F6 0F         RRC
1336 07F9 0F         RRC
1337 07FA 0F         RRC
1338 07FB E60F       SEGC3: ANI    OFH
1339 07FD 12          STAX    D
1340 07FE 1B          DCR     D
1341 07FF C9         RET
1342
1343
1344
1345
1346
1347 0800              ORG     8391H
1348 33F1              USESP: DS    20H
1349 C8B1              RST1:  DS    3
1350 33B4              RST2:  DS    3
1351 C8B7              RST3:  DS    3
1352 33BA              RST4:  DS    3
1353 E3B0              RST5:  DS    3
1354 33B3              RST6:  DS    3
1355 E3C3              RST7:  DS    3
1356 33C6              G01N:  DS    1
1357 E3C9              PORT1: DS    2
1358 33CA              G02N:  DS    1
1359 E3CC              PORT0: DS    2
1360 33CD              D0W1:  DS    1
1361 E3CF              D0W2:  DS    1
1362 33D0              D0W3:  DS    1
1363 E3D1              D0W4:  DS    1
1364 33D2              ADW1:  DS    1
1365 E3D3              ADW2:  DS    1
1366 33D4              ADW3:  DS    1
1367 E3D5              ADW4:  DS    1
1368 33D6              SVDAT: DS    4
1369 E3D7              ENTCT: DS    1
1370 33D8              FTRAP: DS    1
1371 E3DC              FREQ:  DS    1
1372 33DE              PHODE: DS    1
1373 E3DF              SVPH1: DS    2
1374 33E0              SVPH2: DS    2
1375 E3E1              SVPH3: DS    2
1376 33E2              SVPH4: DS    2
1377 E3E3              SVPH5: DS    2
1378 33E4              SVPH6: DS    2
1379 E3E5              SVPH7: DS    2
1380 33E6              SVPH8: DS    2
1381 E3E7              SVPH9: DS    2
1382 33E8              SVPH10: DS    2
1383 E3E9              SVPH11: DS    2
1384 33EA              SVPH12: DS    2
1385 E3EB              SVPH13: DS    2
1386 33EC              SVPH14: DS    2
1387 E3ED              SVPH15: DS    2
1388 33EE              SVPH16: DS    2
1389 E3EF              SVPH17: DS    2
1390 33F0              SVPH18: DS    2
1391 E3F1              SVPH19: DS    2
1392 33F2              SVPH20: DS    2
1393 E3F3              SVPH21: DS    2
1394 33F4              SVPH22: DS    2
1395 E3F5              SVPH23: DS    2
1396 33F6              SVPH24: DS    2
1397 E3F7              SVPH25: DS    2
1398 33F8              SVPH26: DS    2
1399 E3F9              SVPH27: DS    2
1400 33FA              SVPH28: DS    2
1401 E3FB              SVPH29: DS    2
1402 33FC              SVPH30: DS    2
1403 E3FD              SVPH31: DS    2
1404 33FE              SVPH32: DS    2
1405 E3FF              SVPH33: DS    2
1406 3400              SVPH34: DS    2
1407 E401              SVPH35: DS    2
1408 3402              SVPH36: DS    2
1409 E403              SVPH37: DS    2
1410 3404              SVPH38: DS    2
1411 E405              SVPH39: DS    2
1412 3406              SVPH40: DS    2
1413 E407              SVPH41: DS    2
1414 3408              SVPH42: DS    2
1415 E409              SVPH43: DS    2
1416 340A              SVPH44: DS    2
1417 E40B              SVPH45: DS    2
1418 340C              SVPH46: DS    2
1419 E40D              SVPH47: DS    2
1420 340E              SVPH48: DS    2
1421 E40F              SVPH49: DS    2
1422 3410              SVPH50: DS    2
1423 E411              SVPH51: DS    2
1424 3412              SVPH52: DS    2
1425 E413              SVPH53: DS    2
1426 3414              SVPH54: DS    2
1427 E415              SVPH55: DS    2
1428 3416              SVPH56: DS    2
1429 E417              SVPH57: DS    2
1430 3418              SVPH58: DS    2
1431 E419              SVPH59: DS    2
1432 341A              SVPH60: DS    2
1433 E41B              SVPH61: DS    2
1434 341C              SVPH62: DS    2
1435 E41D              SVPH63: DS    2
1436 341E              SVPH64: DS    2
1437 E41F              SVPH65: DS    2
1438 3420              SVPH66: DS    2
1439 E421              SVPH67: DS    2
1440 3422              SVPH68: DS    2
1441 E423              SVPH69: DS    2
1442 3424              SVPH70: DS    2
1443 E425              SVPH71: DS    2
1444 3426              SVPH72: DS    2
1445 E427              SVPH73: DS    2
1446 3428              SVPH74: DS    2
1447 E429              SVPH75: DS    2
1448 342A              SVPH76: DS    2
1449 E42B              SVPH77: DS    2
1450 342C              SVPH78: DS    2
1451 E42D              SVPH79: DS    2
1452 342E              SVPH80: DS    2
1453 E42F              SVPH81: DS    2
1454 3430              SVPH82: DS    2
1455 E431              SVPH83: DS    2
1456 3432              SVPH84: DS    2
1457 E433              SVPH85: DS    2
1458 3434              SVPH86: DS    2
1459 E435              SVPH87: DS    2
1460 3436              SVPH88: DS    2
1461 E437              SVPH89: DS    2
1462 3438              SVPH90: DS    2
1463 E439              SVPH91: DS    2
1464 343A              SVPH92: DS    2
1465 E43B              SVPH93: DS    2
1466 343C              SVPH94: DS    2
1467 E43D              SVPH95: DS    2
1468 343E              SVPH96: DS    2
1469 E43F              SVPH97: DS    2
1470 3440              SVPH98: DS    2
1471 E441              SVPH99: DS    2
1472 3442              SVPH100: DS    2
1473 E443              SVPH101: DS    2
1474 3444              SVPH102: DS    2
1475 E445              SVPH103: DS    2
1476 3446              SVPH104: DS    2
1477 E447              SVPH105: DS    2
1478 3448              SVPH106: DS    2
1479 E449              SVPH107: DS    2
1480 344A              SVPH108: DS    2
1481 E44B              SVPH109: DS    2
1482 344C              SVPH110: DS    2
1483 E44D              SVPH111: DS    2
1484 344E              SVPH112: DS    2
1485 E44F              SVPH113: DS    2
1486 3450              SVPH114: DS    2
1487 E451              SVPH115: DS    2
1488 3452              SVPH116: DS    2
1489 E453              SVPH117: DS    2
1490 3454              SVPH118: DS    2
1491 E455              SVPH119: DS    2
1492 3456              SVPH120: DS    2
1493 E457              SVPH121: DS    2
1494 3458              SVPH122: DS    2
1495 E459              SVPH123: DS    2
1496 345A              SVPH124: DS    2
1497 E45B              SVPH125: DS    2
1498 345C              SVPH126: DS    2
1499 E45D              SVPH127: DS    2
1500 345E              SVPH128: DS    2
1501 E45F              SVPH129: DS    2
1502 3460              SVPH130: DS    2
1503 E461              SVPH131: DS    2
1504 3462              SVPH132: DS    2
1505 E463              SVPH133: DS    2
1506 3464              SVPH134: DS    2
1507 E465              SVPH135: DS    2
1508 3466              SVPH136: DS    2
1509 E467              SVPH137: DS    2
1510 3468              SVPH138: DS    2
1511 E469              SVPH139: DS    2
1512 346A              SVPH140: DS    2
1513 E46B              SVPH141: DS    2
1514 346C              SVPH142: DS    2
1515 E46D              SVPH143: DS    2
1516 346E              SVPH144: DS    2
1517 E46F              SVPH145: DS    2
1518 3470              SVPH146: DS    2
1519 E471              SVPH147: DS    2
1520 3472              SVPH148: DS    2
1521 E473              SVPH149: DS    2
1522 3474              SVPH150: DS    2
1523 E475              SVPH151: DS    2
1524 3476              SVPH152: DS    2
1525 E477              SVPH153: DS    2
1526 3478              SVPH154: DS    2
1527 E479              SVPH155: DS    2
1528 347A              SVPH156: DS    2
1529 E47B              SVPH157: DS    2
1530 347C              SVPH158: DS    2
1531 E47D              SVPH159: DS    2
1532 347E              SVPH160: DS    2
1533 E47F              SVPH161: DS    2
1534 3480              SVPH162: DS    2
1535 E481              SVPH163: DS    2
1536 3482              SVPH164: DS    2
1537 E483              SVPH165: DS    2
1538 3484              SVPH166: DS    2
1539 E485              SVPH167: DS    2
1540 3486              SVPH168: DS    2
1541 E487              SVPH169: DS    2
1542 3488              SVPH170: DS    2
1543 E489              SVPH171: DS    2
1544 348A              SVPH172: DS    2
1545 E48B              SVPH173: DS    2
1546 348C              SVPH174: DS    2
1547 E48D              SVPH175: DS    2
1548 348E              SVPH176: DS    2
1549 E48F              SVPH177: DS    2
1550 3490              SVPH178: DS    2
1551 E491              SVPH179: DS    2
1552 3492              SVPH180: DS    2
1553 E493              SVPH181: DS    2
1554 3494              SVPH182: DS    2
1555 E495              SVPH183: DS    2
1556 3496              SVPH184: DS    2
1557 E497              SVPH185: DS    2
1558 3498              SVPH186: DS    2
1559 E499              SVPH187: DS    2
1560 349A              SVPH188: DS    2
1561 E49B              SVPH189: DS    2
1562 349C              SVPH190: DS    2
1563 E49D              SVPH191: DS    2
1564 349E              SVPH192: DS    2
1565 E49F              SVPH193: DS    2
1566 34A0              SVPH194: DS    2
1567 E4A1              SVPH195: DS    2
1568 34A2              SVPH196: DS    2
1569 E4A3              SVPH197: DS    2
1570 34A4              SVPH198: DS    2
1571 E4A5              SVPH199: DS    2
1572 34A6              SVPH200: DS    2
1573 E4A7              SVPH201: DS    2
1574 34A8              SVPH202: DS    2
1575 E4A9              SVPH203: DS    2
1576 34AA              SVPH204: DS    2
1577 E4AB              SVPH205: DS    2
1578 34AC              SVPH206: DS    2
1579 E4AD              SVPH207: DS    2
1580 34AE              SVPH208: DS    2
1581 E4AF              SVPH209: DS    2
1582 34B0              SVPH210: DS    2
1583 E4B1              SVPH211: DS    2
1584 34B2              SVPH212: DS    2
1585 E4B3              SVPH213: DS    2
1586 34B4              SVPH214: DS    2
1587 E4B5              SVPH215: DS    2
1588 34B6              SVPH216: DS    2
1589 E4B7              SVPH217: DS    2
1590 34B8              SVPH218: DS    2
1591 E4B9              SVPH219: DS    2
1592 34BA              SVPH220: DS    2
1593 E4BB              SVPH221: DS    2
1594 34BC              SVPH222: DS    2
1595 E4BD              SVPH223: DS    2
1596 34BE              SVPH224: DS    2
1597 E4BF              SVPH225: DS    2
1598 34C0              SVPH226: DS    2
1599 E4C1              SVPH227: DS    2
1600 34C2              SVPH228: DS    2
1601 E4C3              SVPH229: DS    2
1602 34C4              SVPH230: DS    2
1603 E4C5              SVPH231: DS    2
1604 34C6              SVPH232: DS    2
1605 E4C7              SVPH233: DS    2
1606 34C8              SVPH234: DS    2
1607 E4C9              SVPH235: DS    2
1608 34CA              SVPH236: DS    2
1609 E4CB              SVPH237: DS    2
1610 34CC              SVPH238: DS    2
1611 E4CD              SVPH239: DS    2
1612 34CE              SVPH240: DS    2
1613 E4CF              SVPH241: DS    2
1614 34D0              SVPH242: DS    2
1615 E4D1              SVPH243: DS    2
1616 34D2              SVPH244: DS    2
1617 E4D3              SVPH245: DS    2
1618 34D4              SVPH246: DS    2
1619 E4D5              SVPH247: DS    2
1620 34D6              SVPH248: DS    2
1621 E4D7              SVPH249: DS    2
1622 34D8              SVPH250: DS    2
1623 E4D9              SVPH251: DS    2
1624 34DA              SVPH252: DS    2
1625 E4DB              SVPH253: DS    2
1626 34DC              SVPH254: DS    2
1627 E4DD              SVPH255: DS    2
1628 34DE              SVPH256: DS    2
1629 E4DF              SVPH257: DS    2
1630 34E0              SVPH258: DS    2
1631 E4E1              SVPH259: DS    2
1632 34E2              SVPH260: DS    2
1633 E4E3              SVPH261: DS    2
1634 34E4              SVPH262: DS    2
1635 E4E5              SVPH263: DS    2
1636 34E6              SVPH264: DS    2
1637 E4E7              SVPH265: DS    2
1638 34E8              SVPH266: DS    2
1639 E4E9              SVPH267: DS    2
1640 34EA              SVPH268: DS    2
1641 E4EB              SVPH269: DS    2
1642 34EC              SVPH270: DS    2
1643 E4ED              SVPH271: DS    2
1644 34EE              SVPH272: DS    2
1645 E4EF              SVPH273: DS    2
1646 34F0              SVPH274: DS    2
1647 E4F1              SVPH275: DS    2
1648 34F2              SVPH276: DS    2
1649 E4F3              SVPH277: DS    2
1650 34F4              SVPH278: DS    2
1651 E4F5              SVPH279: DS    2
1652 34F6              SVPH280: DS    2
1653 E4F7              SVPH281: DS    2
1654 34F8              SVPH282: DS    2
1655 E4F9              SVPH283: DS    2
1656 34FA              SVPH284: DS    2
1657 E4FB              SVPH285: DS    2
1658 34FC              SVPH286: DS    2
1659 E4FD              SVPH287: DS    2
1660 34FE              SVPH288: DS    2
1661 E4FF              SVPH289: DS    2
1662 3500              SVPH290: DS    2
1663 E501              SVPH291: DS    2
1664 3502              SVPH292: DS    2
1665 E503              SVPH293: DS    2
1666 3504              SVPH294: DS    2
1667 E505              SVPH295: DS    2
1668 3506              SVPH296: DS    2
1669 E507              SVPH297: DS    2
1670 3508              SVPH298: DS    2
1671 E509              SVPH299: DS    2
1672 350A              SVPH300: DS    2
1673 E50B              SVPH301: DS    2
1674 350C              SVPH302: DS    2
1675 E50D              SVPH303: DS    2
1676 350E              SVPH304: DS    2
1677 E50F              SVPH305: DS    2
1678 3510              SVPH306: DS    2
1679 E511              SVPH307: DS    2
1680 3512              SVPH308: DS    2
1681 E513              SVPH309: DS    2
1682 3514              SVPH310: DS    2
1683 E515              SVPH311: DS    2
1684 3516              SVPH312: DS    2
1685 E517              SVPH313: DS    2
1686 3518              SVPH314: DS    2
1687 E519              SVPH315: DS    2
1688 351A              SVPH316: DS    2
1689 E51B              SVPH317: DS    2
1690 351C              SVPH318: DS    2
1691 E51D              SVPH319: DS    2
1692 351E              SVPH320: DS    2
1693 E51F              SVPH321: DS    2
1694 3520              SVPH322: DS    2
1695 E521              SVPH323: DS    2
1696 3522              SVPH324: DS    2
1697 E523              SVPH325: DS    2
1698 3524              SVPH326: DS    2
1699 E525              SVPH327: DS    2
1700 3526              SVPH328: DS    2
1701 E527              SVPH329: DS    2
1702 3528              SVPH330: DS    2
1703 E529              SVPH331: DS    2
1704 352A              SVPH332: DS    2
1705 E52B              SVPH333: DS    2
1706 352C              SVPH334: DS    2
1707 E52D              SVPH335: DS    2
1708 352E              SVPH336: DS    2
1709 E52F              SVPH337: DS    2
1710 3530              SVPH338: DS    2
1711 E531              SVPH339: DS    2
1712 3532              SVPH340: DS    2
1713 E533              SVPH341: DS    2
1714 3534              SVPH342: DS    2
1715 E535              SVPH343: DS    2
1716 3536              SVPH344: DS    2
1717 E537              SVPH345: DS    2
1718 3538              SVPH346: DS    2
1719 E539              SVPH347: DS    2
1720 353A              SVPH348: DS    2
1721 E53B              SVPH349: DS    2
1722 353C              SVPH350: DS    2
1723 E53D              SVPH351: DS    2
1724 353E              SVPH352: DS    2
1725 E53F              SVPH353: DS    2
1726 3540              SVPH354: DS    2
1727 E541              SVPH355: DS    2
1728 3542              SVPH356: DS    2
1729 E543              SVPH357: DS    2
1730 3544              SVPH358: DS    2
1731 E545              SVPH359: DS    2
1732 3546              SVPH360: DS    2
1733 E547              SVPH361: DS    2
1734 3548              SVPH362: DS    2
1735 E549              SVPH363: DS    2
1736 354A              SVPH364: DS    2
1737 E54B              SVPH365: DS    2
1738 354C              SVPH366: DS    2
1739 E54D              SVPH367: DS    2
1740 354E              SVPH368: DS    2
1741 E54F              SVPH369: DS    2
1742 3550              SVPH370: DS    2
1743 E551              SVPH371: DS    2
1744 3552              SVPH372: DS    2
1745 E553              SVPH373: DS    2
1746 3554              SVPH374: DS    2
1747 E555              SVPH375: DS    2
1748 3556              SVPH376: DS    2
1749 E557              SVPH377: DS    2
1750 3558              SVPH378: DS    2
1751 E559              SVPH379: DS    2
1752 355A              SVPH380: DS    2
1753 E55B              SVPH381: DS    2
1754 355C              SVPH382: DS    2
1755 E55D              SVPH383: DS    2
1756 355E              SVPH384: DS    2
1757 E55F              SVPH385: DS    2
1758 3560              SVPH386: DS    2
1759 E561              SVPH387: DS    2
1760 3562              SVPH388: DS    2
1761 E563              SVPH389: DS    2
1762 3564              SVPH390: DS    2
1763 E565              SVPH391: DS    2
1764 3566              SVPH392: DS    2
1765 E567              SVPH393: DS    2
1766 3568              SVPH394: DS    2
1767 E569              SVPH395: DS    2
1768 356A              SVPH396: DS    2
1769 E56B              SVPH397: DS    2
1770 356C              SVPH398: DS    2
1771 E56D              SVPH399: DS    2
1772 356E              SVPH400: DS    2
1773 E56F              SVPH401: DS    2
1774 3570              SVPH402: DS    2
1775 E571              SVPH403: DS    2
1776 3572              SVPH404: DS    2
1777 E573              SVPH405: DS    2
1778 3574              SVPH406: DS    2
1779 E575              SVPH407: DS    2
1780 3576              SVPH408: DS    2
1781 E577              SVPH409: DS    2
1782 3578              SVPH410: DS    2
1783 E579              SVPH411: DS    2
1784 357A              SVPH412: DS    2
1785 E57B              SVPH413: DS    2
1786 357C              SVPH414: DS    2
1787 E57D              SVPH415: DS    2
1788 357E              SVPH416: DS    2
1789 E57F              SVPH417: DS    2
1790 3580              SVPH418: DS    2
1791 E581              SVPH419: DS    2
1792 3582              SVPH420: DS    2
1793 E583              SVPH421: DS    2
1794 3584              SVPH422: DS    2
1795 E585              SVPH423: DS    2
1796 3586              SVPH424: DS    2
1797 E587              SVPH425: DS    2
1798 3588              SVPH426: DS    2
1799 E589              SVPH427: DS    2
1800 358A              SVPH428: DS    2
1801 E58B              SVPH429: DS    2
1802 358C              SVPH430: DS    2
1803 E58D              SVPH431: DS    2
1804 358E              SVPH432: DS    2
1805 E58F              SVPH433: DS    2
1806 3590              SVPH434: DS    2
1807 E591              SVPH435: DS    2
1808 3592              SVPH436: DS    2
1809 E593              SVPH437: DS    2
1810 3594              SVPH438: DS    2
1811 E595              SVPH439: DS    2
1812 3596              SVPH440: DS    2
1813 E597              SVPH441: DS    2
1814 3598              SVPH442: DS    2
1815 E599              SVPH443: DS    2
1816 359A              SVPH444: DS    2
1817 E59B              SVPH445: DS    2
1818 359C              SVPH446: DS    2
1819 E59D              SVPH447: DS    2
1820 359E              SVPH448: DS    2
1821 E59F              SVPH449: DS    2
1822 35A0              SVPH450: DS    2
1823 E5A1              SVPH451: DS    2
1824 35A2              SVPH452: DS    2
1825 E5A3              SVPH453: DS    2
1826 35A4              SVPH454: DS    2
1827 E5A5              SVPH455: DS    2
1828 35A6              SVPH456: DS    2
1829 E5A7              SVPH457: DS    2
1830 35A8              SVPH458: DS    2
1831 E5A9              SVPH459: DS    2
1832 35AA              SVPH460: DS    2
1833 E5AB              SVPH461: DS    2
1834 35AC              SVPH462: DS    2
1835 E5AD              SVPH463: DS    2
1836 35AE              SVPH464: DS    2
1837 E5AF              SVPH465: DS    2
1838 35B0              SVPH466: DS    2
1839 E5B1              SVPH467: DS    2
1840 35B2              SVPH468: DS    2
1841 E5B3              SVPH469: DS    2
1842 35B4              SVPH470: DS    2
1843 E5B5              SVPH471: DS    2
1844 35B6              SVPH472: DS    2
1845 E5B7              SVPH473: DS    2
1846 35B8              SVPH474: DS    2
1847 E5B9              SVPH475: DS    2
1848 35BA              SVPH476: DS    2
1849 E5BB              SVPH477: DS    2
1850 35BC              SVPH478: DS    2
1851 E5BD              SVPH479: DS    2
1852 35BE              SVPH480: DS    2
1853 E5BF              SVPH481: DS    2
1854 35C0              SVPH482: DS    2
1855 E5C1              SVPH483: DS    2
1856 35C2              SVPH484: DS    2
1857 E5C3              SVPH485: DS    2
1858 35C4              SVPH486: DS    2
1859 E5C5              SVPH487: DS    2
1860 35C6              SVPH488: DS    2
1861 E5C7              SVPH489: DS    2
1862 35C8              SVPH490: DS    2
1863 E5C9              SVPH491: DS    2
1864 35CA              SVPH492: DS    2
1865 E5CB              SVPH493: DS    2
1866 35CC              SVPH494: DS    2
1867 E5CD              SVPH495: DS    2
1868 35CE              SVPH496: DS    2
1869 E5CF              SVPH497: DS    2
1870 35D0              SVPH498: DS    2
1871 E5D1              SVPH499: DS    2
1872 35D2              SVPH500: DS    2
1873 E5D3              SVPH501: DS    2
1874 35D4              SVPH502: DS    2
1875 E5D5              SVPH503: DS    2
1876 35D6              SVPH504: DS    2
1877 E5D7              SVPH505: DS    2
1878 35D8              SVPH506: DS    2
1879 E5D9              SVPH507: DS    2
1880 35DA              SVPH508: DS    2
1881 E5DB              SVPH509: DS    2
1882 35DC              SVPH510: DS    2
1883 E5DD              SVPH511: DS    2
1884 35DE              SVPH512: DS    2
1885 E5DF              SVPH513: DS    2
1886 35E0              SVPH514: DS    2
1887 E5E1              SVPH515: DS    2
1888 35E2              SVPH516: DS    2
1889 E5E3              SVPH517: DS    2
1890 35E4              SVPH518: DS    2
1891 E5E5              SVPH519: DS    2
1892 35E6              SVPH520: DS    2
1893 E5E7              SVPH521: DS    2
1894 35E8              SVPH522: DS    2
1895 E5E9              SVPH523: DS    2
1896 35EA              SVPH524: DS    2
1897 E5EB              SVPH525: DS    2
1898 35EC              SVPH526: DS    2
1899 E5ED              SVPH527: DS    2
1900 35EE              SVPH528: DS    2
1901 E5EF              SVPH529: DS    2
1902 35F0              SVPH530: DS    2
1903 E5F1              SVPH531: DS    2
1904 35F2              SVPH532: DS    2
1905 E5F3              SVPH533: DS    2
1906 35F4              SVPH534: DS    2
1907 E5F5              SVPH535: DS    2
1908 35F6              SVPH536: DS    2
1909 E5F7              SVPH537: DS    2
1910 35F8              SVPH538: DS    2
1911 E5F9              SVPH539: DS    2
1912 35FA              SVPH540: DS    2
1913 E5FB              SVPH541: DS    2
1914 35FC              SVPH542: DS    2
1915 E5FD              SVPH543: DS    2
1916 35FE              SVPH544: DS    2
1917 E5FF              SVPH545: DS    2
1918 3600              SVPH546: DS    2
1919 E601              SVPH547: DS    2
1920 3602              SVPH548: DS    2
1921 E603              SVPH549: DS    2
1922 3604              SVPH550: DS    2
1923 E605              SVPH551: DS    2
1924 3606              SVPH552: DS    2
1925 E607              SVPH553: DS    2
1926 3608
```


1402 1770	SEC5	EDU	1770H
1404 0031	PS1	EDU	49
1404 0030	PS2	EDU	48
1405 0035	PS3	EDU	53
1406 0037	PS4	EDU	55
1407 001C	PRA	EDU	29
1408 001A	PRB	EDU	26
1407 1748	P10	EDU	193EH
1410 001C	PDA	EDU	28
1411 001D	PDB	EDU	29
1412 0044	PQ0	EDU	48
1413 0020	PQ1	EDU	32
1414 000E	HIGH	EDU	00EH
1415 001F	BIMAX	EDU	31
1416 0006	BIMIN	EDU	8
1417 0020	BOFWD	EDU	32
1418 004A	BOBCK	EDU	74
1419 0035	C503H	EDU	53
1420 0047	DELAY	EDU	71
1421 F125	B5BRK	EDU	0F125H
1422 0000	END		

SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS
--------	------	--------	------	--------	------	--------	------	--------	------

ADRES	93EE	ADRST	0166	ADM1	83DA	ADM2	83D5	ADM3	83D4
ADM4	83D3	ALCLA	00E0	ARSET	017C	AMCH	0451	BOBCK	004A
BOFWD	0020	BIMAX	001F	BIMIN	0009	BRDPT	026F	BRET	025D
B5BRK	F125	BTRUP	0407	BTUP1	0417	BTUP2	0421	C503H	0035
CA	0629	CD	0639	CD12	0643	CH	064E	CHLBC	060C
CKSM1	0561	CKSHO	0402	CLEA1	0042	CLEA2	0043	CHLR	003F
CONPA	03FF	CONT	01A4	CONT1	01C3	CTRLW	0072	DATA	83EC
DOM1	83D2	DMG2	83D1	DMG3	83D0	DMG4	83CF	DELAY	0047
DELE1	07BF	DELEN	07BD	DIG	83F8	DIG1T	00BB	DISP	83F4
DLNCH	066D	DSPOE	03FA	DTCLA	00E3	DTSET	02AB	DMCH	0664
ENTCT	93DB	ENRGLD	0356	ERR	0617	ERRRS	0612	ERROR	037B
FILE	04EF	FIN	02D6	FKEY	83DD	FLAGC	05E9	FLDAD	04DC
FMODE	83DF	FMOV	03A2	FMOV1	0386	FMOV2	03D2	FOUND	0513
FOUT	02FF	FREG	83DE	FSAV1	0439	FSAV2	044B	FSAV3	0480
FSAVE	0425	FTM	0322	FTM1	0339	FTM2	033F	FTM3	0347
FTM4	0355	FTM5	035B	FTRAP	83DC	GDIN	83C9	GGOUT	83CC
HIGH	00DB	INIT	0202	INPRP	0697	INPRT	0690	KEY	068C
KEY1	06DC	KEY2	06E4	KEYIN	0684	KEYND	0055	LOAD1	0534
LSIFT	05F9	MD	00F2	MDPKP	00DA	MDPRD	0169	MODES	00FB
MKEYC	000L	MKEYS	009B	NKEYC	0487	NTEDU	0608	ONE11	07AF
ONEIN	07AC	ONEOT	073B	OUTIN	02F9	P10	193E	PD4	00EF
PC5	00DF	PC6	00BF	PIN	02ED	PDO	0044	PD1	0020
PSA	001C	PQB	001D	PORTA	00F8	PORTC	00FA	PORT1	83CA
PORT0	83CD	PRIT	0319	PRA	001C	PRB	001A	PR100	050E
PRWD	02BF	PRREG	0295	PS1	0031	PS2	0030	PS3	0035
PS4	0637	RDEEC	018D	RDINC	0183	RDCS1	0769	RDCS2	076B
RDSCH	0765	REBER	0AE7	REDR1	0AEC	REG	00E9	REGFN	0112
REGP1	05A6	RESE1	0642	RESE2	064A	RESET	005F	RSDS1	07D7
RSDSP	07CF	RSIFT	029E	RST1	83B1	RST2	83B4	RST3	83B7
RST4	83BA	RSTS	83BD	RSTS5	83C0	RST6	83C3	RST7	83C6
RUN	0197	SCAN	06D4	SECS	1770	SEGC1	07E7	SEGC2	07F7
SEGC3	07FB	SEGGC	07DF	SEGCV	054C	SEGDA	0583	SEGMD	059E
SEGRP	057D	SERCH	07C6	SETXF	0682	SHIFT	05D2	SR111	078B
SR112	079A	SR11N	07B3	SRIOT	0701	SROT1	070C	SROT2	071E
SROT3	0728	START	007C	STAT1	00B2	STAT2	0095	STAT3	009B
STAT1	05F5	STEP	0219	STP1	0226	SVAF	83EA	SVBC	83EB
SVBD	83F2	SVBP	83F0	SVDAT	83D7	SVDE	83EA	SVNL	83EA
SVFC	83E0	SVSP	83E2	TFUNC	00AB	TPDMN	03F0	TRAP	01D3
TRAP1	01E4	TRGAD	0158	TRGMN	0146	USESP	8391	WORDC	0677
WTENT	0279	WVCS1	0299	WVCHG	0257	ZEROT	074A		

10章 モニタサブルーチンの使い方

1. はじめに

通常プログラムは、メインプログラムとそれに付随するいくつかのサブルーチンから構成されています。

サブルーチンプログラムとは、繰り返し行われる処理や共通に使われる処理を一連のプログラムとしてまとめたものです。

モニタプログラムにもいくつかのサブルーチンが存在しています。ここに紹介するサブルーチンは、これらの中でも一般性があり、ユーザーがアプリケーションプログラムを作成する上で、有効であると思われるものです。

TK-80が発売されて以来、これまでに数多くのアプリケーションプログラムが作成されてきました。これらのアプリケーションプログラムがそのまま使えるようにTK-80で公開したRGDSP、SEGCG、INPUTそしてKEYINという四つのサブルーチンをソフトウェアコンパチブルにしています。

2. サブルーチンの考え方

サブルーチンを使用する際に、理解しておかなければならないことがいくつかあります。

その中の一つにスタックの考え方があります。サブルーチンをコールしたとき、サブルーチン処理が終了した場合の戻り番地を記憶しておかなければなりません。この戻り番地はサブルーチンコールされたときに、スタックポインタが指しているRAM上のスタックエリアに自動的に書き込まれ、サブルーチンの終了でRET命令を実行することによって引用されます。

したがって、一般にサブルーチンを用いるときは、あらかじめスタックポインタの設定を行う必要があります。ところが、TK-85では、ユーザープログラムを実行させる際にモニタがスタックポインタを8391H番地にセットしています。したがって特別にスタックポインタの設定を行わなくても、サブルーチンコール等のスタック操作も可能なわけです。

次に考慮すべきことは、パラメータが必要なサブルーチンであるかどうかということです。サブルーチンは一連の処理を行います。処理によってはパラメータが必要となる場合があります。通常ライブラリーとして用意されているサブルーチンではパラメータが当然必要となります。高級言語の場合は、これらパラメータの受け渡しを引き数あるいは共通領域の使用等によって行いますが、機械語を使用する場合は、レジスタ渡し、あるいはRAMを利用して行うことになりま

す、モニタサブルーチンを使用する際は、パラメータが必要な場合には適正な設定を行い、また出力パラメータがある場合は、それをこわすことのないよう注意して下さい。

つぎに考慮すべきことは、モニタサブルーチンの処理内部で破壊されるレジスタがあるということです。したがってこわされては困るレジスタがある場合は、スタックに退避する等の対策を講じなければなりません。

最後にスタックの動作を図10.1によって具体的に説明します。

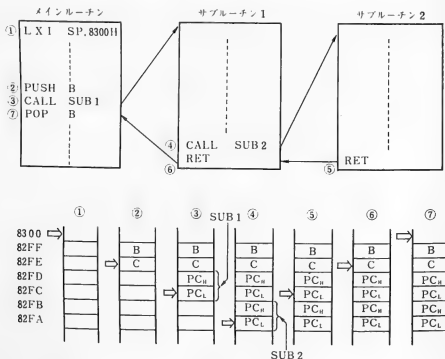


図10.1

ここでちょっとしたテクニックを紹介しましょう。サブルーチン1の内部をみると、サブルーチン2へのコール命令と RET 命令が続いています。このような場合上記2命令を JMP SUB2 と置き換えても同じ動作を行います。JMP SUB2 と置き換えた場合命令バイト数が1バイト減り、スタックレベルも1レベル少なくて済みます。

3. 表示用サブルーチン

TK-85は、モニターエリア内のセグメントデータバッファ(83F8H番地から83FFH番地)の内容をDMA転送によって常時表示しています。

実際にデータをLEDディスプレイに表示させるためには、表示させるデータ(セグメントデータ)を直接上記のエリアに格納すればよいわけです。

なお一つの文字は、1バイト(8ビット)のデータで表示されますが、このデータは各ビットが次のように各セグメントに対応して構成されています。



A : BIT 0	E : BIT 4
B : BIT 1	F : BIT 5
C : BIT 2	G : BIT 6
D : BIT 3	H : BIT 7

セグメントは点灯させるセグメントに対応するビットを“1”とし、点灯させないセグメントに対応するビットを“0”として構成します。

このように、セグメントデータを直接セグメントバッファに格納すればデータは表示されますが、これでは表示させたいデータを基にセグメントデータをそのたびに計算しなければなりません。またプログラム作成の上でもかなり手間のかかることが予想されます。そこでユーザーがプログラムを作成する上で利用できるサブルーチンを公開し、表示処理による手間を省いています。

3.1 表示用サブルーチン構成

モニタには、モニタ処理に便利のように幾つかのサブルーチンが用意してありますが、それぞれのサブルーチンがどのような機能を持っているのか整理する意味で、まず図10.2に示す表示用サブルーチン構成図を理解して下さい。

3.2 機能説明

3.2.1 SEGCV

- | | | |
|------------|---|--|
| (1) スタート番地 | 056CH 番地 | |
| (2) 入出力条件 | 入力パラメータ | モードフラグ(83DFH 番地),
レジスタフラグ(83DEH 番地) |
| | 出力パラメータ | なし |
| | 使用レジスタ | A, F, B, C, D, E, H, L |
| (3) 機 能 | アドレスおよびデータディスプレイレジスタ8バイトに格納されている
ディスプレイレジスタデータを LED の8セグメントデータに変換し、セグメントデータバッファ
に転送します。 | |

したがってあるデータを表示させたい場合には、表示させたいディスプレイレジスタデータをアドレスおよびデータディスプレイエリアにセットしてからこのサブルーチンをコールするだけで済みます。

またユーザーに直接関係はありませんが、モードフラグおよびレジスタフラグの内容をみて、ドットを点灯させる処理を行っています。

各レジスタおよびLEDディスプレイにおけるデータ転送の関係、そしてディスプレイデータ、ディスプレイレジスタデータ、セグメントデータの関係は図10、3のようになっています。

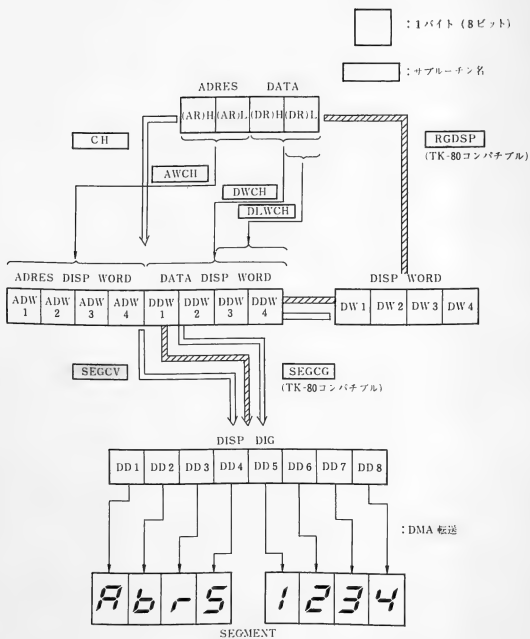


图 10.2

3. 表示用サブルーチン

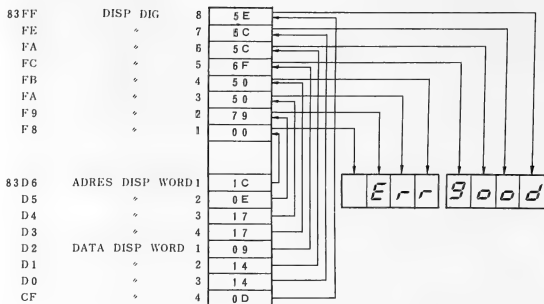
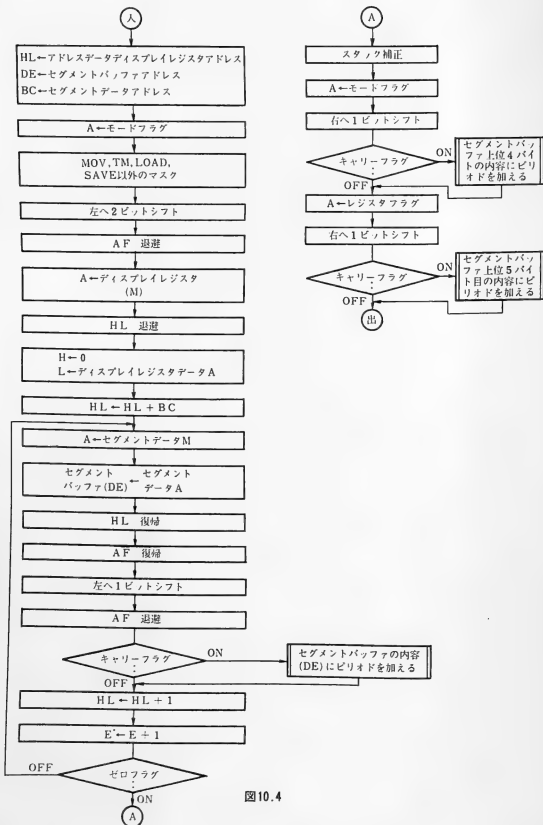


図10.3

ディスプレイデータ	0	1	2	3	4	5	6	7
ディスプレイレジスタデータ	00	01	02	03	04	05	06	07
セグメントデータ	3F	06	5B	4F	66	6D	7D	27
ディスプレイデータ	8	9	A	b	c	d	E	F
ディスプレイレジスタデータ	08	09	0A	0B	0C	0D	0E	0F
セグメントデータ	7F	6F	77	7C	39	5E	79	71
ディスプレイデータ	H	J	L	n	o	P	q	r
ディスプレイレジスタデータ	10	11	12	13	14	15	16	17
セグメントデータ	76	1E	38	54	5C	73	67	50
ディスプレイデータ	U	v	y	h	BLANK	-	.	
ディスプレイレジスタデータ	18	19	1A	1B	1C	1D	1E	
セグメントデータ	3F	1C	6E	74	00	40	80	

注) アドレスおよびデータディスプレイレジスタに1FH以上の値を入れてサブルーチンコールをすると、モニタの命令コードの一部をセグメントデータとみなしてデータ変換を行ってしまいます。

(4) フローチャート 図10.4にフローチャートを示します。



(5) 使用例

8000 ORG 8000H *LEDの*

 MVI B,8 06 08

 LXI H,ADW1 21 D6 83

8005 MVI M,80H 36 1C

 DCX H 2B

 DCR B 05

 JNZ 8005H C2 05 80

800C XRA A AF

 PUSH PSW F5

800E MVI B,7 06 07

 LXI H,ADW2 21 D5 83

 LXI D,ADW1 11 D6 83

8016 MOV A,M 7E

 STAX D 12

 DCX H 2B

 DCX D 1B

 DCR B 05

 JNZ 8016H C2 16 80

 POP PSW F1

 STA DDW4 32 CF 83

 PUSH PSW F5

 CALL SEGCV CD 6C 05

 LXI H,60FFH 21 FF 60

8029 DCR L 2D

 JNZ 8029H C2 29 80

 DCR H 25

 JNZ 8029H C2 29 80

 POP PSW F1

 INR A 3C

 PUSH PSW F5

 CPI 1FH FE 1F

 JNZ 800EH C2 0E 80

 JMP 800CH C3 0C 80

ADW1 EQU 83D6H

ADW2 EQU 83D5H

DDW4 EQU 83CFH

SEGCV EQU 056CH

 END

このプログラムは、モニタの用意したディスプレイデータをLEDの右端から順番に表示させていくプログラムです。無限ループになっているので、まるでネオンサインのように、文字が流れていきます。

3.2.3 RGDSP

- (1) スタート番地 01A1H 番地あるいは 07CFH 番地
- (2) 入出力条件 入力パラメータ モードフラグ (83DFH 番地),
レジスタフラグ (83DEH 番地)
出力パラメータ なし
使用レジスタ A, F, B, C, D, E, H, L
- (3) 機能 アドレスレジスタおよびデータレジスタにセットされているデータをディスプレイレジスタに転送し、さらに各データの上位4ビット、下位4ビットをおのおの16進数として各セグメントデータに変換して、セグメントデータバッファに転送します。

これにより、アドレスレジスタおよびデータレジスタにセットされているデータを、LEDディスプレイに表示させることができます。

各レジスタおよびLEDディスプレイにおけるデータ転送の関係は図10.7のようになっています。

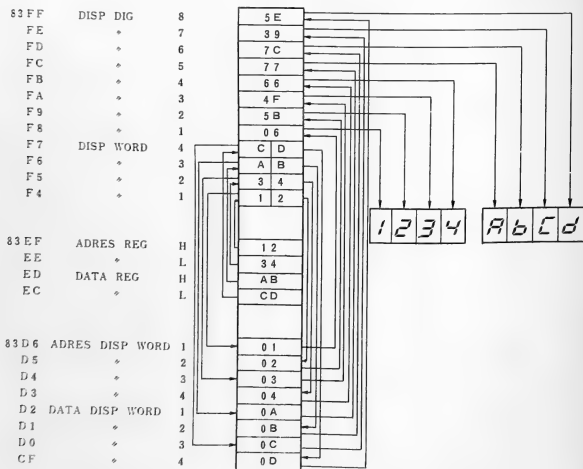


図10.7

3. 表示用サブルーチン

(4) フローチャート 図10.8にフローチャートを示します。

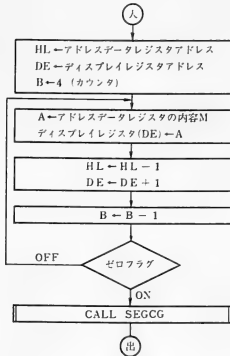


図10.8

(5) 使用例

	ORG	8000H			
8000	LXI	H, 1234H	21	34	12
	SHLD	ADRES	22	EE	83
	LXI	H, 0ABCDH	21	CD	AB
	SHLD	DATA	22	EC	83
	CALL	RGDSP	CD	A1	01
	HLT		76		
ADRES	EQU	83EEH			
DATA	EQU	83ECH			
RGDSP	EQU	01A1H			
	END				

このプログラムはLED表示部に **1234ABCD** を表示します。

3.2.4 DLWCH

- | | | |
|------------|---------|---------------|
| (1) スタート番地 | 066DH番地 | |
| (2) 入出力条件 | 入力パラメータ | なし |
| | 出力パラメータ | なし |
| | 使用レジスタ | A, F, B, H, L |

(3) 機能 データレジスタの下位1バイトの上位4ビット、下位4ビットをおおの16進数とみなして、ディスプレイレジスタデータに変換し、データディスプレイレジスタの下位2ワードに転送します。このサブルーチンは、LEDディスプレイの上位6桁（アドレスディスプレイおよびデータディスプレイの上位2桁）に表示するデータはそのままにしておき、LEDディスプレイの下位2桁（データディスプレイの下位2桁）に表示する16進データのみをデータディスプレイレジスタに転送する用途に使用します。

これは、LEDディスプレイの上位6桁に16進データと他の文字とを混在させて表示し、下位2桁に16進データを表示させようとする場合に便利です。

各レジスタのデータ転送の関係は図10.9のようになっています。

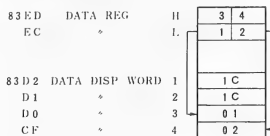


図10.9

(4) フローチャート 図10.10にフローチャートを示します。

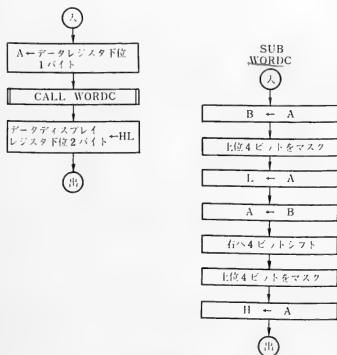


図10.10

(5) 使用例

	ORG	8000H			
8000	MVI	A, 85H	3E	85	
	STA	DATA	32	EC	83
	LXI	H, 0E17H	21	17	0E
	SHLD	ADW2	22	D5	83
	LXI	H, 1714H	21	14	17
	SHLD	ADW4	22	D3	83
	LXI	H, 171CH	21	1C	17
	SHLD	DDW2	22	D1	83
	CALL	DLWCH	CD	6D	06
	CALL	SEGCV	CD	6C	05
	HLT		76		
ADW2	EQU	83D5H			
ADW4	EQU	83D3H			
DDW2	EQU	83D1H			
DLWCH	EQU	066DH			
SEGCV	EQU	056CH			
	END				

このプログラムはLED表示部に `Error 85` を表示します。

3.2.5 DWCH

(1) スタート番地 0664H番地

(2) 入出力条件 入力パラメータ なし

出力パラメータ なし

使用レジスタ A, F, B, H, L

(3) 機能 データレジスタ2バイトの各データ上位4ビット、下位4ビットをおのおの16進数とみなして、ディスプレイレジスタデータに変換し、データディスプレイレジスタ4バイトに転送します。

各レジスタのデータ転送の関係は図10.11のようになっています。

(4) フローチャート 図10.12にフローチャートを示します。

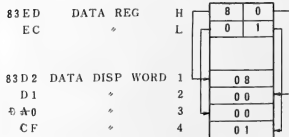


図10.11

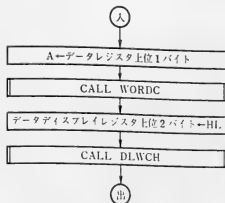


図10.12

(4) 使用例

	ORG	8000H			
8000	LXI	H, 8001H	21	01	80
	SHLD	DATA	22	EC	83
	LXI	H, 1C15H	21	15	1C
	SHLD	ADW2	22	D5	83
	LXI	H, 0C1DH	21	1D	0C
	SHLD	ADW4	22	D3	83
	CALL	DWCH	CD	64	06
	CALL	SEGCV	CD	6C	05
	HLT		76		
DATA	EQU	83ECH			
ADW2	EQU	83D5H			
ADW4	EQU	83D3H			
DWCH	EQU	0664H			
SEGCV	EQU	056CH			
	END				

このプログラムはLED表示部に PC-8001 を表示します。

3.2.6 AWCH

(1) スタート番地 0651H番地

(2) 入出力条件 入力パラメータ なし

出力パラメータ なし

使用レジスタ A, F, B, H, L

(3) 機能 アドレスレジスタ2バイトの各データ上位4ビット, 下位4ビットをおのの16進数とみなしてディスプレイレジスタデータに変換し, アドレスディスプレイレジスタ4バイトに転送します。

各レジスタのデータ転送の関係は図10.13のようになっています。

(4) フローチャート 図10.14にフローチャートを示します。

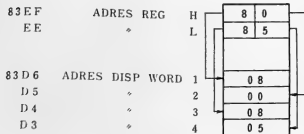


図10.13

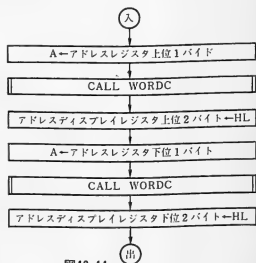


図10.14

(5) 使用例

```

      8000  ORG      8000H
           LXI      H, 8085H    21  85  80
           SHLD     ADRES       22  EE  83
           LXI      H, 0C15H    21  15  0C
           SHLD     DDW2        22  D1  83
           LXI      H, 181CH    21  1C  18
           SHLD     DDW4        22  CF  83
           CALL     AWCH        CD  51  06
           CALL     SEGCV       CD  6C  05
           HLT              76
ADRES EQU 83EEH
DDW2  EQU 83D1H
DDW4  EQU 83CFH
AWCH  EQU 0651H
SEGCV EQU 056CH
END

```

このプログラムはLED表示部に 8085CPU を表示します。

3.2.7 CH

(1) スタート番地 064EH 番地

(2) 入出力条件 入力パラメータ なし

出力パラメータ なし

使用レジスタ A, F, B, H, L

(3) 機能 アドレスおよびデータレジスタ4バイトの各データ上位4ビット, 下位4ビットをおのおの16進数とみなして, ディスプレイレジスタデータに変換し, アドレスおよびデータディスプレイレジスタに転送します。

各レジスタのデータ転送の関係は図10.15のようになっています。

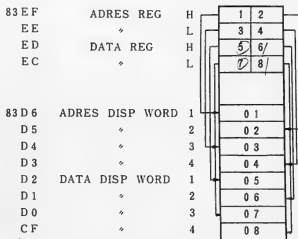


図10.15

- (4) フローチャート 図10.16にフローチャートを示します。

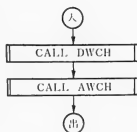


図10.16

- (5) 使用例

	ORG	8000H			
8000	LXI	H, 0	21	00	00
	SHLD	ADRES	22	EE	83
	SHLD	DATA	22	EC	83
8009	CALL	CH	CD	4E	06
	CALL	SEGCV	CD	6C	05
	XRA	A	A7		
8010	DCR	A	3D		
	JNZ	\$-1	C2	10	80
	LHLD	DATA	2A	EC	83
	INX	H	23		
	SHLD	DATA	22	EC	83
	MOV	A, H	7C		
	ANA	A	A7		
	JNZ	8009H	C2	09	80
	MOV	A, L	7D		
	ANA	A	A7		
	JNZ	8009H	C2	09	80
	LHLD	ADRES	2A	EE	83
	INX	H	23		
	SHLD	ADRES	22	EE	83
	JMP	8009H	C3	09	80
ADRES	EQU	83EEH			
DATA	EQU	83ECH			
CH	EQU	064EH			
SEGCV	EQU	056CH			
	END				

このプログラムは、アドレスレジスタおよびデータレジスタを16進カウンタとして使い、それをLED上に表示させるプログラムです。このプログラムも無限ループになっており、アドレスおよびデータレジスタがすべてFFになったら再び0からカウントをやり直します。

4. 入力用サブルーチン

4.1 機能説明

4.1.1 INPUT

- (1) スタート番地 0223H番地あるいは0690H番地
- (2) 入出力条件 入力パラメータ ○キーフラグ (83DDH 番地)
- 00H: 直前のサブルーチンコールでキー入力無
FFH: 直前のサブルーチンコールでキー入力無
- 出力パラメータ ○キー入力無
- A = FFH
キーフラグ = 00H
- キー入力有
- A = 入力データ (16進データ)
キーフラグ = FFH

使用レジスタ A, F, B, D, E, H, L

(3) 機能 キースイッチを8キーずつ三つのブロックに分け、各キーブロックを1回ずつスキャンして、キー入力の有無を調べています。具体的には μ PD8255 (PPI) の PORT A をスキャンデータの入力用に、また PORT C の 4, 5, 6 の3ビットをキーブロックスキャン用に使用しています。各キーブロックをスキャンするときは、各キーブロックに対応した PORT C のビットを Low レベルにすることで、選択を実現しています。選択されたキーブロックの中に押されているキーボードスイッチがあった場合は、対応ビットのみ Low レベルになって、そのキーブロックのデータが PORT A に入力されます。

1 回のキースキャン (3 ブロックすべて) において、いずれのキーボードスイッチも押されない場合は、キーフラグをリセット (00H) し、入力データのパラメータとして用いているアキュムレータに FFH をセットして、このサブルーチンを抜け出します。

1 回目のキースキャンにおいて、キーボードスイッチが押された場合は、チャタリングタイム設定時間 (約 9ms) だけ待ち合わせて、その後再びキースキャンを行います。この2度目のキースキャンにおいて、キー入力が無かった場合 (一度目のキースキャンにおける入力データはノイズ) は、先程と同様にキーフラグをリセット (00H) し、アキュムレータに FFH をセットして、このサブルーチンを抜け出します。

2 回目のキースキャンにおいてキー入力があった場合は、キーフラグをセンスします。これは入力データが今回新たに入力されたものかどうかを判断するためです。したがってこのキーフラグセンスの結果、キーフラグが FFH である状態というのは、直前のサブルーチンコール (INPUT) における入力データと今回の入力データとがまったく同じものであることを意味しています。つ

まりずっとキーボードスイッチが押し続けられている状態を示しています。この場合はキースキャンを引き続き行い、キーボードスイッチが一度離されるまでこのサブルーチン内で待ち続けます。

キーフラグセンスの結果、キーボードスイッチが今回始めて押されたことが検出された場合は、アキュムレータに押されたキーボードスイッチに対応する16進データを格納し、またキーフラグをリセットした後、このサブルーチンを抜け出します。

このサブルーチンを利用すれば、プログラムに何らかの実行をさせながらキーの状態をモニタすることができます。

なおキーボードスイッチと変換される16進データとの関係は表10.1のようになっています。

表10.1

スキャン ポート 入出力	PC-4		PC-5		PC-6	
	入力キー	16進データ	入力キー	16進データ	入力キー	16進データ
PA-0	0 / AF	00	8 /	08	RUN	10
PA-1	1 / BC	01	9 /	09	CONT	11
PA-2	2 / DE	02	A / SAVE	0A	ADRES SET	12
PA-3	3 / HL	03	B / LOAD	0B	READ DEC	13
PA-4	4 / SP	04	C / T M	0C	READ INC	14
PA-5	5 / BR.P	05	D / MOV	0D	WR/ENT	15
PA-6	6 / BR.D	06	E / OUT	0E	MODE	16
PA-7	7 /	07	F / I N	0F	REG	17

キースキャンに付けられている名前は、モニタプログラムで解釈される意味に合わせてありますが、キーの位置とその意味付けはプログラムの処理のみで決まるものですから、同じキーに違った意味を持たせて使用することもできます。

注) チャタリング

メカニカルタイプのスイッチではその操作時に必ずチャタリングが生じます。チャタリ

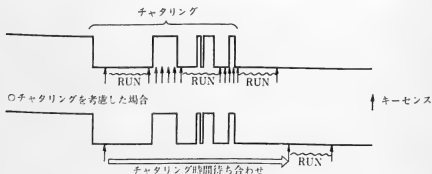


図10.17

4. 入力用サブルーチン

ングを考慮しないでキースキャンを行うと一度しかキースイッチを押さないのに、2度以上押したように動作を行ってしまう可能性があります。このサブルーチンでは、チャタリングを考慮して、一度キー入力検出されたら、チャタリングの時間を待ち合わせてから再びキースキャンを行っています（図10.17参照）。

(4) フローチャート 図10.18にフローチャートを示します。

キ-入力無(FFH) キ-フジラ
キ-入力有(FFH) キ-フジラ

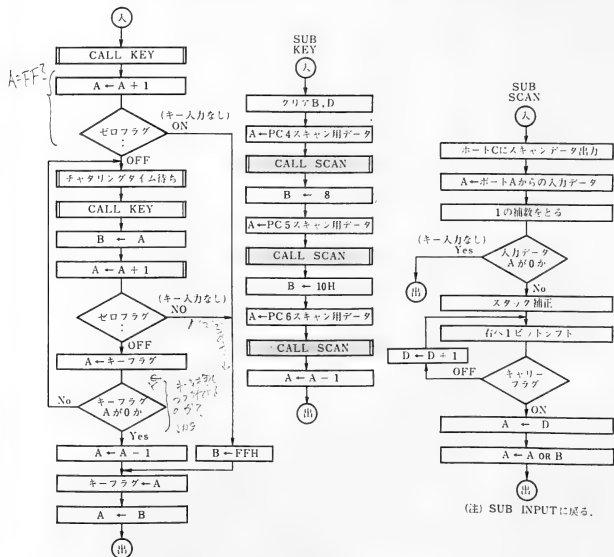


図10.18

(5) 使用例

	ORG	8000H			
8000	LXI	H, 101DH	21	1D	10
	SHLD	DDW2	22	D1	83
	LXI	H, 0	21	00	00
	SHLD	ADRES	22	EE	83
800C	CALL	INPUT	CD	23	02
	STA	DATA	32	EC	83
	INR	A	3C		
	JZ	801DH	CA	1D	30
	LHLD	ADRES	2A	EE	83
	INX	H	23		
	SHLD	ADRES	22	EE	83
801D	CALL	AWCH	CD	51	06
	CALL	DLWCH	CD	6D	06
	CALL	SEGCV	CD	6C	05
	JMP	800CH	C3	0C	80
DDW2	EQU	83D1H			
DATA	EQU	83ECH			
ADRES	EQU	83EEH			
AWCH	EQU	0651H			
DLWCH	EQU	066DH			
SEGCV	EQU	056CH			
INPUT	EQU	0223H			
	END				

このプログラムは、入力したキースイッチの16進データをデータディスプレイ下位2桁に表示し、なおキー入力のカウントを行いそれをアドレスディスプレイ4桁に表示するプログラムです。RUNさせた状態ではLEDには 0000H-FF と表示されます。たとえば次にREGキーを押してみます。カウンタは1増加され、アドレスディスプレイ上には 0001 と表示されます。またREGキーを押している間はデータディスプレイの下位2桁にはREGキーの16進データである 17 が表示されます。

4.1.2 KEYIN

(1) スタート番地 0216H番地あるいは0684H番地

(2) 入出力条件 入力パラメータ ○キーフラグ (83DDH番地)

00H: 直前のサブルーチンコールでキー入力無

FFH: 直前のサブルーチンコールでキー入力有

出力パラメータ ○キー入力無

A=FFH

キーフラグ=00H

○キー入力有

A=入力データ (16進データ)

キーフラグ=FFH

使用レジスタ

A, F, B, D, E, H, L

5. シリアルデータ入出力用サブルーチン

(3) 機能 このキー入力サブルーチンにおいては、キーボードスイッチが始めて押されたことが検出されるまで、このルーチンの中でキースキャンを繰り返します。サブルーチン INPUT と違うところは、サブルーチン INPUT はキースイッチが押されなくてもアキュムレータにデータ FFH を格納してルーチンから抜けてきたのに対し、このキー入力サブルーチンでは新たなキー入力か施されるまでは、キースキャンをサブルーチン内部で繰り返しているという点です。

(4) フローチャート 図10.19にフローチャートを示します。

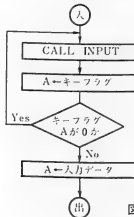


図 10.19

5. シリアルデータ入出力用サブルーチン

5.1 シリアルデータ出力用サブルーチン機能説明

5.1.1 REDER

(1) スタート番地 06E7H 番地

(2) 入出力条件 入力パラメータ なし

出力パラメータ なし

使用レジスタ A, F, D, E

(3) 機能 2,400Hz の信号が約 5 秒間送出されます。カセットテープへの録音 FORMAT で 2,400Hz の波、2 周期分をビットの“1”と定義しました。したがってこの FORMAT で説明すると、ビット“1”を 6,000 ビット送出するルーチンということになります。モニタの CMT SAVE において、ファイルヘッドを生成するのにこのルーチンを用いています (図10.20参照)。

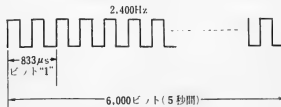


図 10.20

(4) フローチャート 図10.21にフローチャートを示します。

(5) 使用例 モニタサブルーチン SRIOT参照

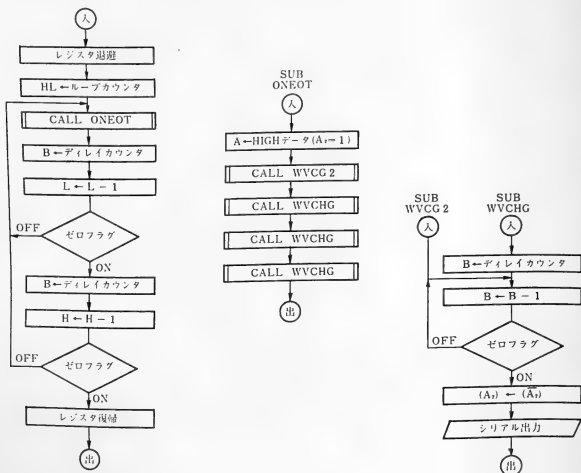


図10.21

5.1.2 SRIOT

- (1) スタート番地 0701H番地
- (2) 入出力条件 入力パラメータ A = 出力データ
B = デレイカウンタ
出力パラメータ なし
使用レジスタ A, F

(3) 機能 アキュムレータに格納されている1バイトデータを、下記 FORMAT に従ってシリアルデータに変換して SOD 端子から出力します。シリアルデータは、周波数 2,400Hz の波 2 周期をデータ "1" とし、周波数 1,200Hz の波 1 周期をデータ "0" としています。

5. シリアルデータ入出力用サブルーチン

○データ転送速度 1,200baud (bit/s)

○1ビットデータ形式

"0" ...1,200Hz 1サイクル

"1" ...2,400Hz 2サイクル



○1バイトデータ形式 (1バイトデータ73Hの場合): SOD 出力波形

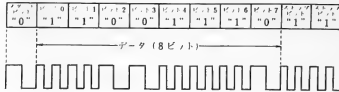


図10.22

ハードウェアの簡略化を図るため、カセットテープへの出力波形はすべてソフトウェアで生成しています。したがってユーザーがこのルーチンを使用する場合にはいくつかの法則に従わなければなりません。

1バイトデータを出力する場合は、このルーチンを呼ぶことによって波形生成は自動的に行われます。しかし、READER 出力ルーチンを呼んだ後1バイト出力ルーチンを呼ぶとか、1バイト出力ルーチンを呼んだ後再び1バイト出力ルーチンを呼ぶという、いわゆる接続部分はユーザーが管理しなければなりません。この接続部分のし寄せはすべて2番目に呼ばれる1バイト出力ルーチンのスタートビットが請負います。この接続部分、つまりスタートビットの波形を正常な大ききで生成するためには1バイト出力ルーチンを呼ぶ前に正常な遅延時間を作る適当なパラメータをBレジスタに設定しなければなりません。

では適切なパラメータの設定方法を二つの場合に分けて説明しましょう。

① READER 出力ルーチンをコールした後、1バイト出力ルーチンをコールする場合の設定方法

n : Bレジスタに設定する遅延パラメータ (整数)

X : READER 出力ルーチン後1バイト出力ルーチンをコールするまでに要したステート数。ただし、1バイト出力ルーチンコール命令も含む。

$$n = \frac{1}{14} (809 - X)$$

② 1バイト出力ルーチンをコールした後、再び1バイト出力ルーチンをコールする場合の設定方法

n : Bレジスタに設定する遅延パラメータ (整数)

X : 1バイト出力ルーチンコール後、再び1バイト出力ルーチンをコールするまでに要したステート数。ただし、2番目の1バイト出力ルーチンコール命令も含む。

$$n = \frac{1}{14} (852 - X)$$

(4) フローチャート 図10.23にフローチャートを示します。

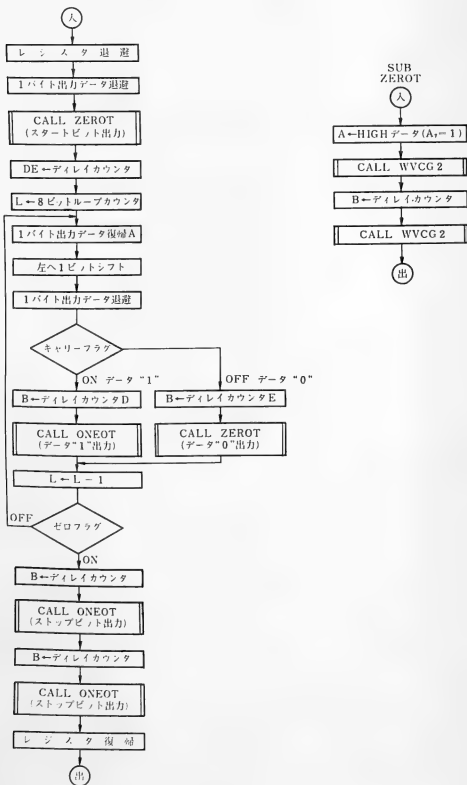


図10.23

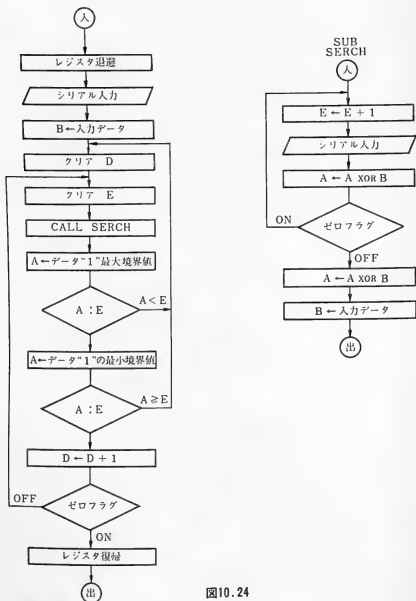


図10.24

5.2.2 **SRIIN**

(1) スタート番地 0783H番地

(2) 入出力条件 入力パラメータ なし
 出力パラメータ A = 入力データ
 使用レジスタ A, F

(3) 機能 SID端子より入力されるシリアルデータを受信して8ビットデータに編集して、アキュムレータに格納します。

1バイトデータのFORMATは1バイトデータ出力サブルーチンで述べた通りです。したがってこのサブルーチンに入るとまずスタートビットの検出を行います。

5. シリアルデータ入出力用サブルーチン

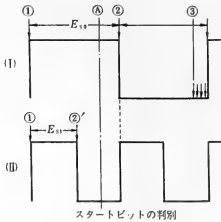


図10.25

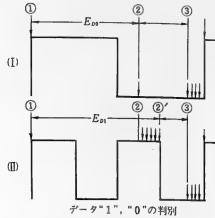


図10.26

図10.25において(I)はスタートビットの場合、(II)はREADER部の場合です。まず①の変化点が発出されると、カウンタEがクリアされ再び変化点があるまでカウントアップを始めます。(I)の場合は②の変化点までカウントアップされます。②までカウントアップされたカウンタを E_{s0} とします。(II)の場合は②'の変化点までカウントアップされます。この場合のカウンタの値を E_{s1} とします。図から明らかなように、 $E_{s0} > E_{s1}$ となっています。したがってこれらの値の間に両者を見分ける境界値を設けて、この境界値(④)とカウンタとを比較すれば、スタートビットかどうかの判別はつくわけです。

(I)の場合のように、スタートビットが発出されると、その後②から③の位置まで遅延動作を行ない再び変化点検出のスクアンを始めます。この遅延動作はスクアンを行う時期をできるだけ次の変化点に近づけ、ノイズによる誤判定を防ぐためのものです。

スタートビット判別後は、シリアルデータの読み込みを行い、1バイトデータに編集しなければなりません。ここでは編集以前の問題であるデータの判別方法を述べます。

図10.26においてスタートビット判別方法と同様に①の変化点においてカウンタEはクリアされます。その後、②の時点まではカウンタEのカウントアップだけを行い変化点検出は行いません。②の時点まできて始めて、変化点検出を始めます。データが"0"の場合はこの時点でデータが変化しているわけですから、この値がそのままカウンタの値 E_{00} となります。ところがデータが"1"の場合はその後もカウントアップを続け、②にきて始めてカウントアップを終了します。したがってカウンタの値をみれば②の時点でカウントを終了したものか、あるいはその後もカウントアップを続けたものであるかは一目瞭然です。

データ判別後再び次のデータの判別に移るわけですが、この場合も③の時点まで遅延動作を行い、ノイズによる誤判定を防いでいます。

このSRIINルーチンを用いるときに注意することは、SRIINルーチンを複数回コールする場合、それらの間が4,000ステート以上にならないことです。4,000ステート以上の時間が経過すると、次の1バイトデータのスタートビットが始まってしまい、データの読み込みが間に合わなく

なってしまいます。

(4) フローチャート 図10.27にフローチャートを示します。

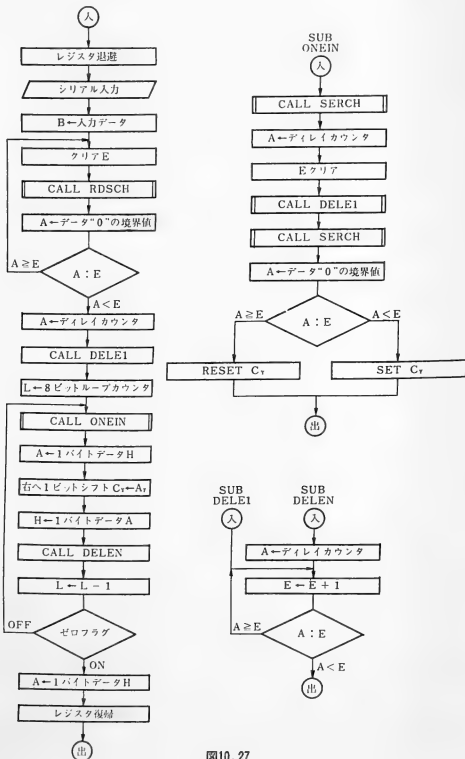


図10.27

5. シリアルデータ入出力用サブルーチン

(5) 使用例

	ORG	8000H			
8000	LXI	H, 8100H	21	00	81
	MVI	B, 0	06	00	
	CALL	RDSCH	CD	65	07
8008	CALL	SRIIN	CD	83	07
	MOV	M, A	77		
	INX	H	23		
	INR	B	04		
	JNZ	8008H	C2	08	80
	HLT		76		
RDSCH	EQU	0765H			
SRIIN	EQU	0783H			
	END				

READER以降次々と転送されてくるデータを、8100H番地から81FFH番地までに格納します。

11章 TK-85のハードウェア

1. はじめに

本章には、TK-85の回路を機能別に分割し、その個々の機能ブロックについてシステム内での役割割りおよびその動作についての説明が述べられています。

2. TK-85 システム構成

TK-85 は、CPU システムとして必要最小機能により構成されており、また今後 TK-85 を機能拡張するための必要なデータバス、アドレスバスおよびコントロール信号がカードエッジに出力されています。

まず、TK-85 のシステムブロック図によりシステム全体の概要を頭に入れ、これから個々の

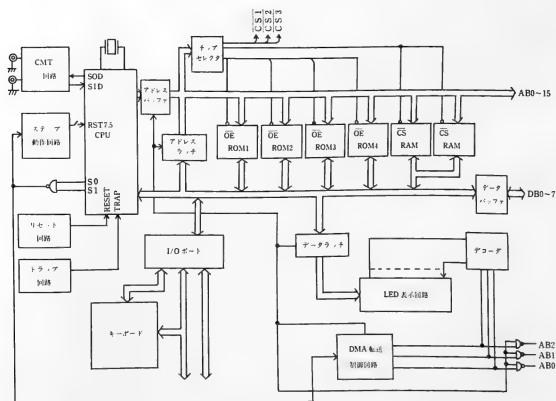


図11.1 システムブロック図

項目で述べられる回路のシステムにおける位置付けを理解して下さい。図11. 1にTK-85のシステムブロック図を示します。

TK-85は大きく分けて、CPU部、メモリ部、キー入力部、表示部、CMT インターフェイス部の五つのシステムブロックにより構成されています。CPU部は、 μ PD8085A CPUを中心として構成されており、システム全体を制御する心臓部であるといえます。

またTK-85のCPUは、約2.5MHzのクロックで動作しています。メモリ部には、ICメモリが使用されており、ROM (Read Only Memory) およびRAM (Random Access Memory) が搭載されています。ROMにはモニタプログラムが格納されており、電源投入時およびRESETキーあるいはMONキーが押されたときに走りはじめます。RAMは、1kバイト搭載されており、一部をモニタプログラムがワーキングエリアとして使用しているほか、プログラムやデータエリアとして使用することができます。

キー入力部は、その制御用にプログラマブル周辺インターフェイス LSI (μ PD8255AC-5) が使われており、これによりTK-85に対するプログラミングおよびその実行、デバッグ等を行う24個のキーボードスイッチがスキャンされます。

表示部は8個の7セグメントLED (発光ダイオード) ディスプレイで構成されており、16進数によりアドレスやデータを表示することができます。表示される内容はプログラムによる転送ではなく、サイクルスチールによるDMA転送によって、常時自動的に転送・表示が行われています。

CMT インターフェイス回路は μ PD8085Aのシリアル入出力端子を利用してデータの入出力を行っています。この回路により市販のオーディオテープレコーダを外部記憶装置として用いることができ、自分の作成したプログラムおよびデータをセーブおよびロードすることができます。

下記にTK-85の仕様を示します。

◀TK-85仕様▶

CPU : μ PD8085A

クロック周波数: 2.4576MHz (4.9152MHzクリスタル使用)

ROM : μ PD2316EC (μ PD2716Dピンコンパチブル)

※実装2kバイト (MAX. 8kバイト)

RAM : μ PD2114LC 1kバイト

入力装置: キーボードスイッチ 25個

表示装置: 8桁7セグメントLEDによる16進数表示

パラレル I/O ポート: μ PD8255AC-5 の8ビットおよび4ビットポートをカードエッジに出
力 (入力, 出力プログラム可能)

バス: TK-80 バス (上位コンパチブル)

CMT 転送レート：1200ボ-

電 源：外部電源が必要 +5V \pm 5% 単一電源

消費電流：1.2A以下

寸 法：310 \times 230mm

3. アドレスバス、データバス

TK-85 で使用されています μ PD8085A はマルチプレクスの方式のデータバスを採用しています。

μ PD8085A のアドレス信号は、上位 8 ビットが直接 CPU の端子より出力されており、下位 8 ビットはデータバス上に CPU がデータの入出力を行っていないタイミングに出力されます。

データバス上に CPU が下位アドレスを出力している期間中、CPU のコントロール信号 ALE は High となり下位アドレスを出力中であることを示します。この下位アドレスは μ PB8212 にこのタイミングでラッチされます。

また上位アドレスは 74LS367 (トライステートバッファ) でドライブされています。TK-85 のアドレスバスはこの 74LS367 より出力される上位アドレスと μ PB8212 より出力される下位アドレスにより形成されています。このうち A 0, A 1, A 2, は 7407 (オープンコレクタバッファ) を介して出力されています。この理由は 7 節で述べます。

7 節で述べる DMA 転送 (Direct Memory Access) が行われる期間、アドレスバスはフローティング状態になります。ここで A 0, A 1, A 2 のバッファ 7407 は入力がハイ インピーダンスによる出力不安定となるのを防ぐため入力側ラインは 1 k Ω の抵抗によりプルアップされています。

TK-85 のアドレスバスは、HALT 時および HOLD 時にフローティング状態にはなりませんので (DMA が行われているため)、外部装置において DMA 転送等を行う場合は、アドレスバスに外部装置の入力側でトライステートバッファを入れる必要があります。TK-85 のデータバスは、 μ PB8216C (オプション) によりドライブされカードエッジに出力されています。このため、外部拡張を行うためには、IC 24 および IC 25 の IC ソケットに μ PB8216C を挿入する必要があります。

データバスは、TK-85 内の ROM, RAM および I/O が選択された場合および RESET 時、HALT 時および HOLD 時に外部とは遮断状態になります。

・ボード上に μ PB8216 を 2 個挿入する場合、IC の向きに注意して下さい。

4. ROM, RAM の構成

CPU システムにおいて ROM, RAM はプログラムやデータの記憶および CPU レジスタ内容の一時退避などに用いられ重要な構成要素となっています。

図11.2 は TK-85 の ROM, RAM 回路部分を示しています。ROMはボード上最大 8 k バイト実装が可能で、 μ PD2716 が 4 個使用できます。TK-85 ではすでに 0 H ~ 7 FFH 番地のエリアにモニタプログラムが常駐するようにマスク ROM μ PD2316EC (μ PD2716 ピンコンパチブル) が実装済みです。

RAM は μ PD2114LC (スタティック) が 2 個実装されていて、1 k バイトの容量です。

ROM, RAM のそれぞれの IC はチップセレクト端子 (μ PD2316 では CS3, μ PD2716 では \overline{OE} 端子) が Low レベルになることにより選択されます。このチップセレクト信号は CPU より出力される IO/\overline{M} , \overline{RD} , \overline{WR} と、ROM, RAM に入力されない残りのアドレス信号、それぞれ 5 本・6 本とを組み合わせて作られます。74LS139 によって出力されるアドレスデコードされたチップ

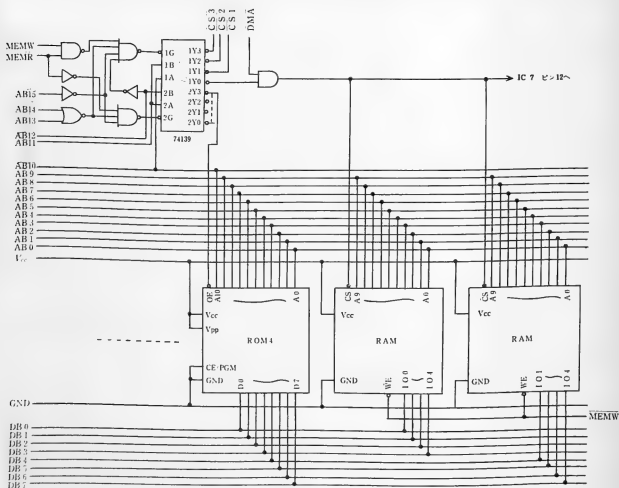


図11.2 ROM, RAM 回路

セレクト信号のうち、8400H～87FFH, 8800H～8BFFH, 8C00～8FFFHを示す、チップセレクト信号 ($\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$) は基板のカードエッジを通して外部へ出力されます。この信号を用いて RAM 増設をすることができます。(12節参照)

また別に外部で ROM, RAM および I/O を増設する場合、外部で使用するコントロール用信号もカードエッジに設けてあります。

5. リセット回路

TK-85 ではパワーオンと同時に CPU に対して RESET 信号が発生し、CPU は RESET 信号解除後に 0 番地より書き込まれているモニタプログラムを実行しはじめます。この動作を行うのが図11. 3で示すリセット回路です。また図11. 5で示すタイミングチャートを見て下さい。

図11. 5のAで示される電圧変化によって電源が投入された場合、CPU の $\overline{\text{RESET IN}}$ 端子に加えられる信号変化をBで示してあります。このBの信号変化はリセット回路の R_2 と C_1 が持つ時定数によりAで示される電源電圧が積分されたものです。CPU の $\overline{\text{RESET IN}}$ の内部にはシュミット回路が組み込まれており、入力されるBの信号波形は CPU 内ではCで示される信号として検出されます。つまり、図11. 5のタイミングチャートにおいて電源投入時TよりT'までの

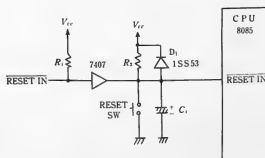


図11. 3 リセット回路

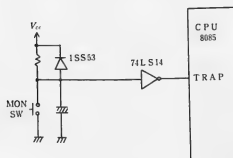


図11. 4 トラップ回路

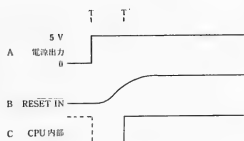


図11.5 電源 ON 時リセット回路タイミングチャート

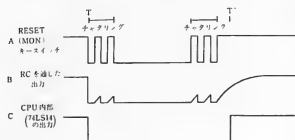


図11.6 RESET (MON) スイッチ ON 時回路タイミングチャート

期間CPUはリセット状態であることになります。これによりCPUはリセット解除後、モニタプログラムを確実に実行しはじめます(この動作をパワーオンリセットと呼びます)。

また、プログラム実行中任意にリセットを行うためにリセット回路にはRESETスイッチが付いています。RESETスイッチを押すことによりCPUのRESET IN端子に入力される信号は強制的にLowレベルになります。

RESETスイッチによりリセット信号を発生させた場合、スイッチより出力される信号は図11.6で示されるタイミングチャートのAと同様なものとなります。Aにおいてスイッチを押した直後、およびスイッチを離れた直後にチャタリングという現象が発生します。チャタリングを含むリセット信号が直接CPUに入力された場合、RESETスイッチを1回押すごとにリセット信号が複数回CPUに入力されることになってしまいます。この現象を防ぐためにも抵抗 R_2 とコンデンサ C_1 が働いています。

図11.6のBはスイッチの出力がCRの積分回路を通して出てくる信号波形です。RESETスイッチを押した直後と離れた直後のチャタリングはCRの時定数によりスレッショールドレベルまで達せずCPU内部では信号変化として検出されません。このようなCRの働きによりRESETスイッチを1回押すことにより1発のリセット信号を発生させることができます。

また、このリセット回路にはカードエッジより入力されるRESET IN信号によっても、CPUに対するリセット信号を発生させられるようになっています。

6. トラップ回路

TK-85のトラップ回路はMON キースイッチが押された場合、CPU がどのようなプログラムを処理していてもモニタプログラムに戻るようにするための回路です。図11. 4はTK-85で使用されているトラップ回路です。リセット回路と同様の構成となっていますが、ただ一つ違う点はシュミット回路として74LS14 がCPU に外付けとなっていることです。

μ PD8085A の TRAP 端子は割り込みを CPU に対して要求するための端子の一つで、ノンマスクابلインターラプト（ソフトウェアにより禁止できない割り込み）となっています。CPU が TRAP 要求を受け付けると、プログラム実行はリスタートアドレス24Hに飛び、次に TRAP 処理プログラムを実行した後、モニタに戻ります。

7. 表示回路と DMA 転送

TK-85において、人間と情報をやり取りする窓口としてデータ・命令を入力する役目を果しているのがキーボードスイッチであり、CPU よりのデータ表示出力をする役目を果しているのが7セグメント LED 8個です。この LED ディスプレイを含むデータの各制御および動作を行うための回路が表示回路です。ここで、RAM より直接表示データを表示回路へ送る方法として DMA 転送が行われています。

TK-85のDMA転送は、サイクルスチールによって行われており、このためにCPUを一時停止状態（HOLD状態）にすることは行っておりません。

CPU μ PD8085が命令を実行してゆく過程において、必ずある一定期間データバスもアドレスバスもCPUが使用しないタイミングが存在します。

これはCPU タイミングのM1（マシンサイクル1）のT4ステートに当たります。

TK-85では、このタイミングをハードウェアにより検出して、このときにメモリと表示回路との中でDMA転送しています。

図11. 7はTK-85で使用されている表示回路の部分を示しています。表示回路においてDMA転送時にシステムを制御するための基本となる信号がゲートG1より出力されるDMACです。この信号はプログラム実行中、OPコードフェッチサイクル*時ごとに出力されています。

図11. 8にOPコードフェッチサイクル時のDMA転送タイミングチャートが示してあります。この図においてDMAC信号は \overline{RD} の信号が立ち上がった後Lowになり、次のALE信号およびM1サイクル信号が出力されると同時にHighに戻ります。DMAC信号がLowである期間CPUはOPコードを、解析、内部実行を行っており、CPUは外部の回路に対してアクセスしていない期間となります。つまりCPUが外部状況を見ていない期間に表示回路はアドレスバ

* μ COM-85ユーザーズマニュアル（IEM-618B）参照

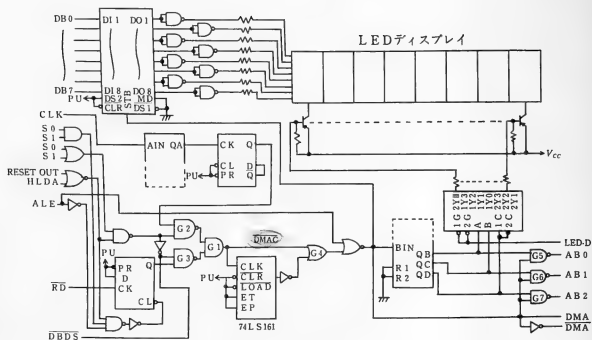


图11.7 表示回路

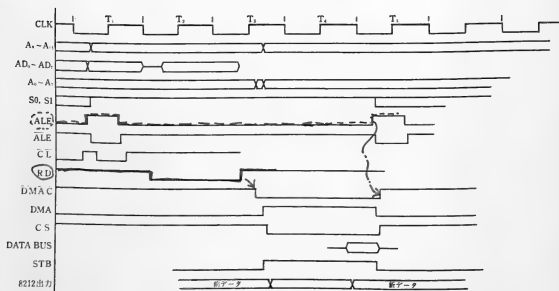


図11.8 DMA 転送 タイミングチャート

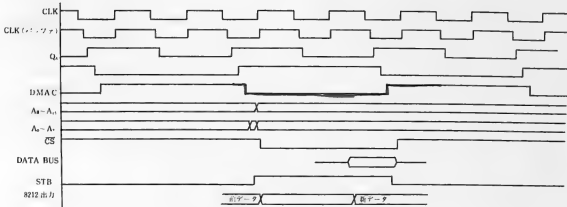


図11.9 DMA 転送 (RESET, HOLD)

ス、データバスを利用して表示データを RAM より得ます。この動作をサイクルスチールというわけです。

プログラム実行中の DMAC 信号はゲート G 3 およびゲート G 1 を通って出力されますが、CPU が RESET, HOLD, HALT 状態の場合、DMAC 信号はゲート G 2 およびゲート G 1 を通って出力されます。この DMAC 信号は CPU クロック (CLK 信号) を 4 分周したものが使用されています。図 11.9 にこのときのタイミングチャートを示します。

ゲート G 1 より出力される DMAC 信号は、次に 74LS161 により 16 分周され出てくる 16 回に 1 回の Carry 信号とともにゲート G 4 に入力します。G 4 より出てくる DMA 信号とそれを反転させた DMA 信号が実際に各回路の制御信号として利用されています。DMA 信号はカウンタに入力され、そのカウンタにより DMA 転送時の RAM アドレスの下位 3 ビットデータを作り出しています。このアドレス下位 3 ビットデータは DMA 転送開始と同時にゲート G 5, G 6, G 7 より出力されます。このとき OP コードフェッチ時のアドレスデータが μ PB8212 よりまだ出力されている可能性がありこの場合、DMA 転送用アドレスデータとぶつかることになってしまいます。このため、 μ PB8212 より出力されているアドレスの下位 3 ビットは 3 節で述べられているようにオープンコレクタバッファ 7407 でドライブされており、データのぶつかり合いが起ってもシステムを破壊しないようになっています。

DMA 転送用アドレスは、アドレスバスの AB0H~AB9H までプルアップされており、DMA および DMA の各信号により制御され、ハイインピーダンス状態になったときに下位 3 ビット以外は High レベルとなります。また DMA 転送時に TK-85 ボード上の実装 RAM が選択されますので、プルアップと組み合わせさせてアドレスが 83F8H~83FFH まで選択されることになります。

8. μ PD8255 とキーボード回路

TK-85 では、I/O ポートとしてプログラマブル周辺インターフェイス μ PD8255 を備えています。 μ PD8255 はプログラム可能な 8 ビット I/O ポートを三つもち、モード 0 ~ 2 の使い分けができます。モード 0 では基本的な入出力ポートとして動作します。モード 1 ではデータの入出力制御にコントロール信号、ステータス信号を用いるモードです。モード 2 ではデータの入力、出力に同一ポートを使用し双方向の入出力ポートとして使用することができます。データ入出力の制御はモード 1 と同様にコントロール信号およびステータス信号を用います。

TK-85 のモニタプログラムでは、 μ PD8255 をモード 0 としてプログラムし、キーボードスイッチのデータ入力用としてポート A とポート C の一部を使用しています。またポート C の PC7 は 9 節で述べます 1 インストラクションステップ動作回路の制御に使われ、残りのポートはカードエッジに接続されています。

μ PD8255 はモニタプログラムによりポート A のビットを入力状態に、またポート C の上位 4 ビットを出力状態にしています。このポート A の 8 ビットとポート C の上位 3 ビットのラインはマトリクス構造となっていて、その各交点に図 11.10 で示すようにキースイッチが結線されています。

ポート A の 8 ビットラインは抵抗 $33\text{ k}\Omega$ によりプルアップされています。キースキャン方法として TK-85 ではポート C の PC4, PC5, PC6 の各ビットに順次 Low レベルを出力します、ポ

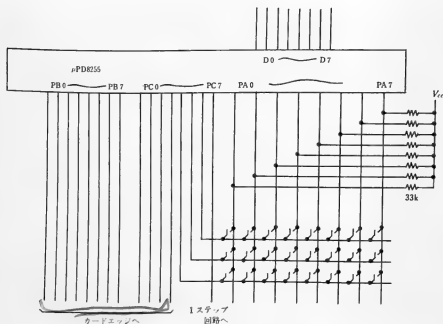


図 11.10 キーボード回路

9. 1 インストラクションステップ動作回路

ートCの各ビットを変化させるごとにポートAのデータをCPUに読み込みます。キースイッチがどれも押されない限りポートAの8ビットラインは High レベルであり、入力データは FFH となります。またキースイッチが押された場合、押されたキースイッチに当るポートAのビットが Low レベルとなるため FFH とはなりません。つまり Low レベルにしたポートCのビットと、そのとき読みとったポートAのデータをみてやることにより、どのキースイッチが押されたかが判断できます。

9. 1 インストラクションステップ動作回路

TK-85 では自作プログラムの解析、検討およびプログラムミスの発見を容易にするために1インストラクションステップ動作回路が備えてあります。

TK-85 の1インストラクションステップ動作回路では割り込みを利用して動作を行い、モニタプログラムにより途中レジスタおよびメモリの内容を読み出し、変更できるようになっています。この項目ではハードウェアの動作について述べます。

まずモニタプログラムより RUN キーおよび CONT キーが押されユーザープログラムにジャンプする部分のプログラムを下記に示します。

```

1.  CONT:  XRA    A
2.         OUT   PORTC
3.         DCR   A
4.         STA   FTRAP
5.         MVI   A, MSKRS
6.         SIM
7.         OUT   PORTC
8.         POP   PSW
9.         EI
10.        RET
  
```

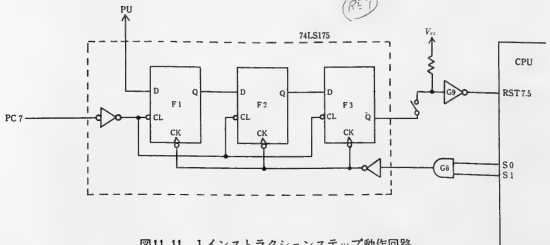
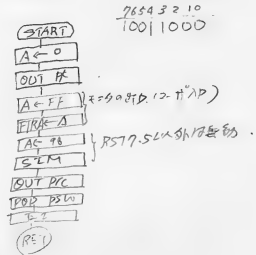


図11.11 1 インストラクションステップ動作回路

ここで CPU が

1. XRA
2. OUT PORTC

の2命令を実行するとポートCの最上位ビット、7ビット目がLowとなり図11.11において各フリップフロップにクリアがかかります。ここでスライドスイッチをSTEP側に倒してある場合、TRAP端子にはLowが入力されます。

以下、次のポート動作時までクリアは解除されません。下記で示す命令以下のCPU動作を図11.11とタイミングチャート図11.12を用いて述べます。

7. OUT PORTC

この命令を実行することによりフリップフロップのクリアは解除され、ゲートG8より出力されるOPコードフェッチサイクルをカウントしはじめます。図11.12の1ステップ動作タイミングチャートを見ると、クリア解除後各フリップフロップはOPコードフェッチ信号を入力するたびに順番に出力が変化してゆき最後に時間TにおいてフリップフロップF3の \bar{Q} がHighからLowに変化します。

フリップフロップF3のQはインバータを介してCPUのRST7.5入力に接続されています。RST7.5は、割り込み要求端子でこの入力信号の立ち上がりのエッジで、割り込み要求が受け付けられます。つまり図11.12でわかるように、割り込みはRET命令の後ジャンプした先の最初の命令の実行が始まった直後に発生することになります。

CPUは、この第1番目の命令実行後割り込み処理モニタにジャンプしてきます。またCONTキーを押すことによって、同様な動作を再び行い割り込み処理モニタにジャンプしてきます。

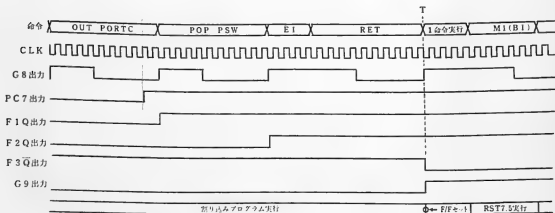


図11.12 1インストラクションステップ動作タイミングチャート

10. CMT 回路

TK-85 ではプログラムやデータを記憶する装置として市販オーディオテープレコーダを利用できます。このテープレコーダと CPU とのインターフェイスとして CMT 回路があります。

CMT 入力および出力回路はそれぞれ出力される信号を相手側に入力しやすいレベルおよび波形にします。

図11.13の(b)に CMT 出力回路を示します。 $\mu\text{PD8085A}$ のシリアル出力端子 SOD より出力される信号は C_2 と R_1 により積分されます。次に R_2 と R_3 によりテープレコーダのマイクロホン端子に入力できるレベルまで分圧しミニプラグに出力します。

図11.13の(a)は CMT 入力回路です。この回路では、市販オーディオテープレコーダより入力されるアナログ波形を整形しデジタル波形にしなければなりません。このため入力信号は保護抵抗 R_4 を通った後、ダイオード 2 個によりダイオードの順方向電圧 0.8V までスライスされます。スライスされた信号は次段のアンプにより増幅されます。増幅された信号はコンパレータ (μPC311) により基準電圧と比較され、比較結果として出力に +5V と 0V のデジタル信号が出力されます。このデジタル信号は CPU の SID 端子に入力され、正確に 0 と 1 の判断を CPU が行います。

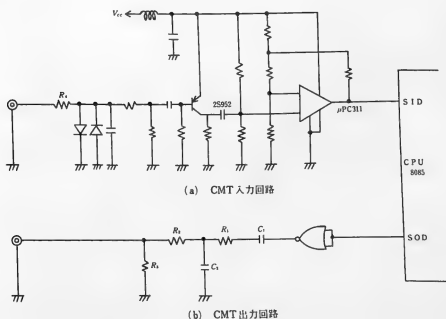


図11.13 CMT回路

11. カードエッジ信号表

ピン	A	B	ピン	A	B	ピン	A	B
1	GND	GND	18			35	PB 5	RST 6.5
2	GND	GND	19			36	PB 6	NC
3	+5 V	+5 V	20		MEMR	37	PB 7	NC
4		NC	21		MEMW	38	PC 0	NC
5		NC	22	READY		39	PC 1	NC
6	ALE	NC	23			40	PC 2	NC
7		NC	24			41	PC 3	NC
8	RD	NC	25	HOLD	HLDA	42	CS 3	INTR
9	WR	IO/M	26		DB 7	43	CS 2	INTR
10	AB15	AB 7	27	DMA	DB 6	44	CS 1	RESET OUT
11	AB14	AB 6	28	DMA	DB 5	45	RESET IN	RESET OUT
12	AB13	AB 5	29	DBSL	DB 4	46		
13	AB12	AB 4	30	PB 0	DB 3	47		
14	AB11	AB 3	31	PB 1	DB 2	48	CLK	S0・S1
15	AB10	AB 2	32	PB 2	DB 1	49		S0+S1
16	AB 9	AB 1	33	PB 3	DB 0	50	GND	GND
17	AB 8	AB 0	34	PB 4	RST 5.5			

<各信号名の説明>

ALE	μPD8085A よりのコントロール信号 (負論理)
RD	μPD8085A よりのコントロール信号 (負論理)
WR	μPD8085A よりのコントロール信号 (負論理)
IO/M	μPD8085A よりのコントロール信号
HLDA	μPD8085A よりのコントロール信号 (負論理)
RESET OUT }	μPD8085A よりのコントロール信号
RESET OUT }	
INTA	μPD8085A よりのコントロール信号 (負論理)
CLK	μPD8085A よりのコントロール信号 (正論理)
READY	μPD8085A の入力信号 (正論理)
RESET IN	μPD8085A の入力信号 (負論理)
HOLD	μPD8085A の入力信号 (負論理)
RST 5.5	μPD8085A の入力信号 (負論理)
RST 6.5	μPD8085A の入力信号 (負論理)
INTR	μPD8085A の入力信号 (負論理)
AB15~AB 0	アドレス信号 (バッファを介して出力)
DB 7~DB 0	データ信号 (μPB8216 を介して出力) (注 μPB8216 を挿入しなければ出力されません)
PB 7~PB 0 }	μPD8255A C-5 の入出力ポート
PC 3~PC 1 }	
CS 3~CS 1	RAM増設用チップセレクト信号
DMA }	DMA 転送期間を示す信号
DMA }	
DBSL	外部データバスが選ばれている期間を示す信号 (負論理)
S0・S1	CPU の OP コードフェッチサイクルを示す (正論理)
S0+S1	CPU の HALT 状態を示す (負論理)
+5 V	電源 (+5 V) ライン
GND	グランドライン
NC	(TK-85 では使用されておりません)

12. RAM 増設

TK-85 にはモニタプログラムのワーキングエリアと自作プログラムエリアとして最小の RAM 1k バイトが実装されています。しかし自作プログラムの容量が増えるにつれて、RAM を増設したくなるでしょう。そのため TK-85 では 74LS139 より出力されるアドレスデコード信号をカードエッジに接続しており、外部でこの信号を利用して簡単に 3 k バイト RAM 増設ができるように考慮されています。

カードエッジより出力される $\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$ はそれぞれ次のように 1 k バイトごとにアドレスデコードしたものです。

$\overline{CS1}$: 8400H ~ 87FFH

$\overline{CS2}$: 8800H ~ 8BFFH

$\overline{CS3}$: 8C00H ~ 8FFFH

増設は 1 k バイトごとに可能です。図 11.14 は $\mu PD2114LC$ を用いた増設回路例です。この場合外部に増設するため TK-85 にオプションの $\mu PB8216$ を実装して下さい。

また、外部に ROM, RAM を数多く増設する場合、アドレスバス、データバスをバッファを用いて強化してから増設を行ってください。

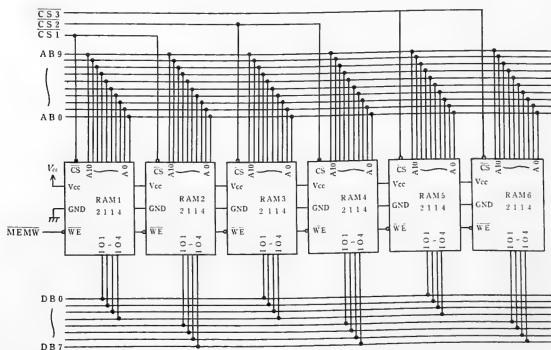


図 11.14 RAM 増設例

付 録

1. μ PD8085ACデータシート

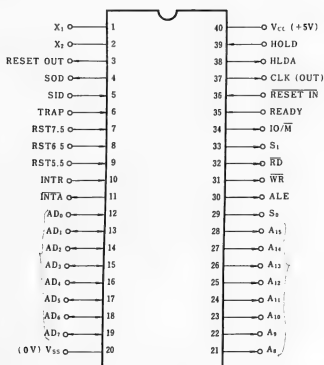
μ PD8085Aは、 μ PD8080AFとソフトウェア・コンパチブルな8 BIT CPUで、 μ PD8080AFシステムよりシステムの高速性を増し、3チップ： μ PD8085A(CPU)、 μ PD8155/8156(RAM+I/O)、 μ PD8355/8755A^{*}(ROM/PROM+I/O)で最小システムを構成できます。 μ PD8085Aは、 μ PD8080AFに必要な μ PB8224(クロック・ジェネレータ)、 μ PB8228(システム・コントローラ)の総ての機能を含み、システムの集積度を上げています。

μ PD8085Aは、マルチプレクス方式のデータバスを採用しており、アドレスの下位8ビットは8 BITのアドレス/データ・バスより出力されますが、 μ PD8155/8156/8355/8755Aメモリのオンチップ・アドレス・ラッチにより、 μ PD8085Aと直接インタフェース可能です。

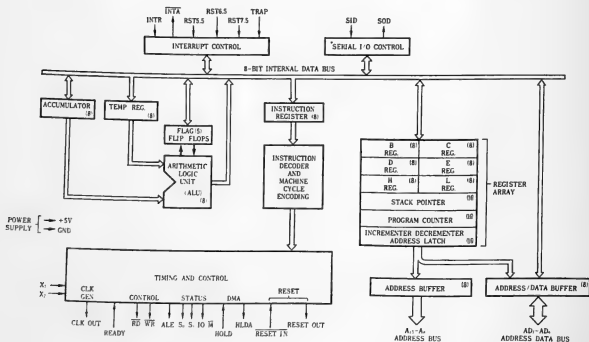
特 徴

- μ COM-85
- 単電源 +5V
- μ PD8080AFと100%ソフトウェア・コンパチブル
- インストラクション・サイクル1.3 μ s (μ PD8085A) ; 0.8 μ s (μ PD8085A-2)
- オンチップC.G. (Ex XTAL or RC)
- オンチップ・システム・コントローラ
- 1本のノンベクトル割込入力
- 4本のベクトル付割込入力 (1本はNon Mask)
- シリアルイン/シリアルアウト・ポート
- 10進、2進、2倍精度演算
- 64Kまで直接アドレス可能
- NチャネルMOS
- 40ピン・プラスチックDIP (μ PD8085AC, μ PD8085AC-2)
- 40ピン・セラミックDIP (μ PD8085AD, μ PD8085AD-2)
- Intel8085A/8085A-2コンパチブル

端子接続図 (Top View)



ブロック図



端子機能

(1) A_8-A_{15} (Address Bus) …… 3ステート出力

メモリアドレスの上位8ビットまたはI/O用8ビットアドレスとなります。ホールドとホールドモード中はハイ・インビデンスとなります。

(2) AD_0-AD_7 (Multiplexed Address/Data Bus) …… 3ステート入出力

最初のクロックサイクルでメモリアドレスの下位8ビット (またはI/Oアドレス) を出力します。2, 3番目のクロックサイクルでは双方向の8ビット・データバスとなります。

ホールドとホールドモード中はハイ・インビデンスとなります。

(3) ALE (Address Latch Enable) …… 出力

最初のクロックで発生し、 AD_0-7 のアドレス信号を周辺チップ内のラッチにラッチ入力するために用いられます。ALEの立下りエッジでアドレス情報のセット・ホールド時間が規定されます。

ALEはステータス情報のストロブ信号としても用いられます。

(4) S_0, S_1 (Data Bus Status) …… 出力

次に示すバスサイクルのエンコードされたステータスが出力されます。

S_1	S_0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

S_1 はアドバンストR/Wステータスとしても用いられます。

(5) \overline{RD} (Read) …… 3ステート出力

選択されたメモリまたはI/Oが読出され、データバスがデータ転送のために使用できることを示します。

ホールドとホールドモード中はハイ・インビデンスとなります。

(6) \overline{WR} (Write) …… 3ステート出力

データバス上のデータを選択されたメモリまたはI/Oに書込むために用いられます。

データのセット時間は \overline{WR} の後縁より規定されます。

ホールドとホールドモード中はハイ・インビデンスとなります。

(7) READY …… 入力

リードまたはライトサイクルにおいてREADY入力がハイならば、メモリまたは周辺チップがデータの送出または受信の準備ができていることを示します。

もしREADY入力がロウならば、CPUはリードまたはライトサイクルを延長し、READYがハイになるのを待ちます。

(8) HOLD …… 入力

別のマスタがアドレスとデータバスの使用を要求していることを示します。CPUはホールド要求を受けたら、そのときのマシンサイクル終了後、ただちにバスの使用をやめます。ただし内部処理は続けられます。CPUはホールド要求がなくなって始めてバスを使用できるようになります。ホールド要求が受けられると、アドレス、データ、 \overline{RD} 、 \overline{WR} 、そしてIO/Mの各ラインがハイ・インビデンスとなります。

(9) HLDA (Hold Acknowledge) ……出力

CPUがホールド要求を受け付け、次のクロックサイクルからバスの使用をやめることを示します。HLDAはホールド要求がなくなった後ロウになります。

CPUはHLDAがロウになった $\frac{1}{2}$ クロックサイクル後にバスの使用を開始します。

(10) INTR (Interrupt Request) ……入力

汎用割込入力として用いられます。INTRは命令の最後から2番目のクロックサイクルでのみ検出されます。もしアクティブであれば、PCのインクリメントは禁止され、 $\overline{\text{INTA}}$ が発生されます。

割込サービスルーチンにジャンプさせるため、このサイクル間にRESTARTまたはCALLを挿入することができます。

INTRはソフトウェアで許可、禁止を制御できます。さらに外部リセットとこの割込が受け付けられた直後に禁止されます。

(11) $\overline{\text{INTA}}$ (Interrupt Acknowledge) ……出力

この信号はINTRが受け付けられた後の命令サイクルでRDの代りに（そしてRDと同じタイミングで）用いられます。これは8259割込チップまたは他の割込ポートをアクティブにするために用いることができます。

(12) RST 5.5, RST 6.5, RST 7.5 (Restart Interrupts) ……入力

これらの3つの入力は自動的に挿入される内部RESTARTを起こすことを除けば、INTRと同じタイミングを持っています。

これらの割込みの優先順位を次に示します。またこれらの割込みの優先順位はINTRより高くなっています。

RST 7.5	最高優先
RST 6.5	↑
RST 5.5	最低優先

(13) TRAP (Trap Interrupt) ……入力

トラップ割込みはノンマスクابل・リスタート割込みです。

これはINTRと同じタイミングで検出されます。この割込みは、いかなるマスクまたは割込イネーブル制御によっても影響を受けません。この割込みは他のどの割込みよりも優先順位が高くなっています。

(14) RESET IN ……入力

リセットによりPCがゼロにされ、割込イネーブルとHLDAフリップフロップがリセットされます。他のフラグやレジスタ（インストラクション・レジスタを除く）はリセットの影響を受けません。リセットが入力されている間、CPUはリセット状態を続けます。リセットが解除されると0番地からプログラムをスタートします。

(15) RESET OUT ……出力

CPUがリセット中であることを示す信号で、システムリセット信号として用いることができます。

この信号はCPUクロックに同期化されています。

(16) X_1 , X_2 ……入力

内部クロック発生回路用のクリスタルまたはRC回路を接続します。さらに X_1 は外部クロック入力端子として用いられます。

(17) CLK (Clock) ……出力

クリスタルまたはCR回路によってCPUクロックを発生させている場合に、システムクロック用のクロック出力として用います。

(18) IO/ \overline{M} (IO/Memory) ……3ステート出力

この信号がハイならばリード/ライトがI/Oに対して行われ、ロウならばメモリに対して行われることを示します。

ホールドとホールドモード中はハイ・インビデンスとなります。

(19) SID (Serial Input Data) ……入力

RIM命令が実行されたとき、このラインのデータがアキュムレータのビット7にロードされます。シリアルデータのインタフェースに使用されます。

(20) SOD (Serial Output Data) ……出力

この出力はSIM命令によってセット、リセットされます。

シリアルデータのインタフェースに使用されます。

割込みとシリアルI/O

8085Aは5本の割込入力(INTR, RST 5.5, RST 6.5, RST 7.5そしてTRAP)を持っています。INTRは8080AのINTと同一機能になっています。RST 5.5, 6.5, 7.5およびTRAPはリスタート割込みですが、TRAPはノンマスク、他の3本はプログラムでマスク可能です。

RST 5.5, 6.5, 7.5の割込入力がいネーブルになったとき、インタラプト・マスクがセットされていなければ、PCの内容をスタックに退避させ、リスタートアドレスに分歧します。TRAP割込みがアクティブになると、インタラプト・イネーブルやマスクに関係なくリスタートアドレスに分歧します。

割込名称	リスタートアドレス (16進)
TRAP	24
RST 5.5	2C
RST 6.5	34
RST 7.5	3C

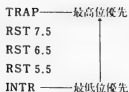
RST 5.5とRST 6.5はINTR(そして8080AのINT)と同様に、ハイレベル検出になっており、INTRと同じタイミングで受け付けられます。RST 7.5は立上りエッジ検出になっています。

RST 7.5に対しては、内部割込要求を発生させるための内部F/Fをセットするのに1個のバルスだけで十分です。RST 7.5リクエストF/Fは、そのリクエストが処理されるまでセットし続け、処理された後自動的にリセットされます。このF/FはSIM命令または外部リセット入力(RESET IN)によってもリセットされます。

なおRST 7.5内部F/Fは、RST 7.5がマスクされているときでもRST 7.5端子に入力された1個のバルスによってセットされます。

3個のRSTインタラプト・マスクはSIM命令またはRESET INによってのみ影響されます。

1個以上の割込みが同時に起こったとき、どれを受け付けるかを決定するために、各割込みに対して次に示す優先順位が定められています。

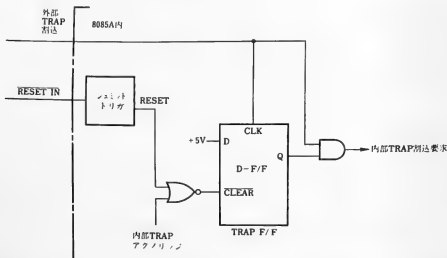


この優先機構は、より高い優先の割込みによってスタートされていたルーチンの優先順位は考慮していません。例えば、RST 5.5は、RST 7.5ルーチンの終了前に各割込みがイネーブルにされていれば、RST 7.5ルーチンに割込むことができます。

TRAP割込みは、電源故障やバスエラーなどに有効です。TRAP割込みは他の割込みと同様に受け付けられますが、優先順位は最も高くなくとも、TRAP割込みは、どのフラグ、マスクによっても影響されません。TRAP入力はエッジおよびレベルの両方で検出されます。

TRAP入力が受け付けられるためには、ハイに立上り、その後ハイを保持しなければなりません、一度ロウに下がってそれからハイに上がるまでは、再び受け付けられることはありません。これによりノイズやロジックグリッチによる誤トリガを避けることができます。

次図に8085A内のTRAP割込要求回路を示します。



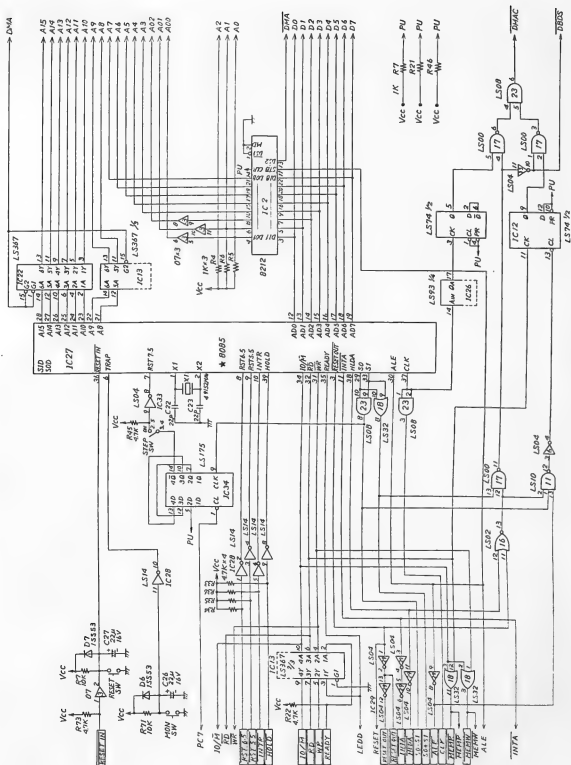
どの割込み (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) でも一度その割込み処理に入ると、EI命令が実行されるまではその後に入力されるすべての割込み (TRAPを除いて) を禁止するということに注意して下さい。

TRAP割込みは、他の割込みがイネーブル状態でもそうでなくても割込み発生し、他の割込みを禁止させることができます。

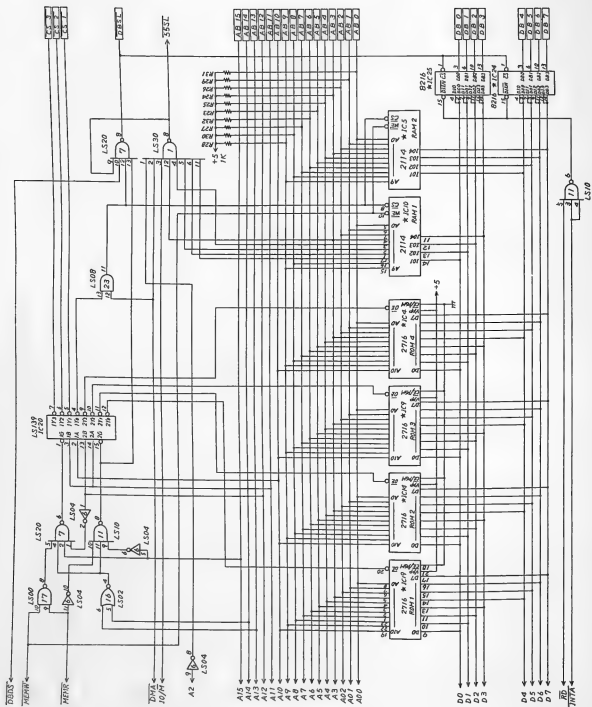
TRAP割込み処理に入った後、RIM命令を実行することによって以前のインタラプト・イネーブル・フラグの状態を知ることができます。

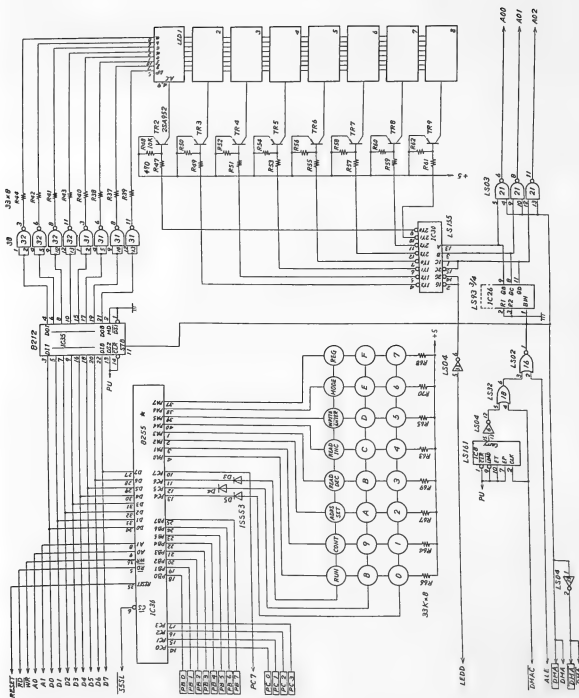
シリアル I/O システムもまた RIM と SIM 命令によって制御されます。SID は RIM 命令によって読まれ、SOD データは SIM 命令によってセットされます。

2. TK-85 全回路図



8085 CPU 返回流程图





**LED,
井一卡一之回控图**

μCOM-85 インストラクション活用表

μCOM-85 機械語 ↔ ニーモニック 対照表

00	NOP		40	MOV	B, B	80	ADD	B	C0	RNZ	
01	LXI	B, B ₁ B ₂	41	MOV	B, C	81	ADD	C	C1	POP	B
02	STAX	B	42	MOV	B, D	82	ADD	D	C2	JNZ	B ₁ B ₂
03	INX	B	43	MOV	B, E	83	ADD	E	C3	JMP	B ₁ B ₂
04	INR	B	44	MOV	B, H	84	ADD	H	C4	CNZ	B ₁ B ₂
05	DCR	B	45	MOV	B, L	85	ADD	L	C5	PUSH	B
06	MVI	B, B ₂	46	MOV	B, M	86	ADD	M	C6	ADI	B ₂
07	RLC		47	MOV	B, A	87	ADD	A	C7	RST	0
08	...		48	MOV	C, B	88	ADC	B	C8	RZ	
09	DAD	B	49	MOV	C, C	89	ADC	C	C9	RET	
0A	LDAX	B	4A	MOV	C, D	8A	ADC	D	CA	JZ	B ₁ B ₂
0B	DCX	B	4B	MOV	C, E	8B	ADC	E	CB	...	
0C	INR	C	4C	MOV	C, H	8C	ADC	H	CC	CZ	B ₁ B ₂
0D	DCR	C	4D	MOV	C, L	8D	ADC	L	CD	CALL	B ₁ B ₂
0E	MVI	C, B ₂	4E	MOV	C, M	8E	ADC	M	CE	ACI	B ₂
0F	RRC		4F	MOV	C, A	8F	ADC	A	CF	RST	1
10	...		50	MOV	D, B	90	SUB	B	D0	RNC	
11	LXI	D, B ₁ B ₂	51	MOV	D, C	91	SUB	C	D1	POP	D
12	STAX	D	52	MOV	D, D	92	SUB	D	D2	JNC	B ₁ B ₂
13	INX	D	53	MOV	D, E	93	SUB	E	D3	OUT	B ₂
14	INR	D	54	MOV	D, H	94	SUB	H	D4	CNC	B ₁ B ₂
15	DCR	D	55	MOV	D, L	95	SUB	L	D5	PUSH	D
16	MVI	D, B ₂	56	MOV	D, M	96	SUB	M	D6	SUI	B ₂
17	RAL		57	MOV	D, A	97	SUB	A	D7	RST	2
18	...		58	MOV	E, B	98	SBB	B	D8	RC	
19	DAD	D	59	MOV	E, C	99	SBB	C	D9	...	
1A	LDAX	D	5A	MOV	E, D	9A	SBB	D	DA	JC	B ₁ B ₂
1B	DCX	D	5B	MOV	E, E	9B	SBB	E	DB	IN	B ₂
1C	INR	E	5C	MOV	E, H	9C	SBB	H	DC	CC	B ₁ B ₂
1D	DCR	E	5D	MOV	E, L	9D	SBB	L	DD	...	
1E	MVI	E, B ₂	5E	MOV	E, M	9E	SBB	M	DE	SBI	B ₂
1F	RAR		5F	MOV	E, A	9F	SBB	A	DF	RST	3
20	RIM		60	MOV	H, B	A0	ANA	B	E0	RPO	
21	LXI	H, B ₁ B ₂	61	MOV	H, C	A1	ANA	C	E1	POP	H
22	SHLD	B ₁ B ₂	62	MOV	H, D	A2	ANA	D	E2	JPO	B ₁ B ₂
23	INX	H	63	MOV	H, E	A3	ANA	E	E3	XTHL	
24	INR	H	64	MOV	H, H	A4	ANA	H	E4	CPO	B ₁ B ₂
25	DCR	H	65	MOV	H, L	A5	ANA	L	E5	PUSH	H
26	MVI	H, B ₂	66	MOV	H, M	A6	ANA	M	E6	ANI	B ₂
27	DAA		67	MOV	H, A	A7	ANA	A	E7	RST	4
28	...		68	MOV	L, B	A8	XRA	B	E8	RPE	
29	DAD	H	69	MOV	L, C	A9	XRA	C	E9	PCHL	
2A	LHLD	B ₁ B ₂	6A	MOV	L, D	AA	XRA	D	EA	JPE	B ₁ B ₂
2B	DCX	H	6B	MOV	L, E	AB	XRA	E	EB	XCHG	
2C	INR	L	6C	MOV	L, H	AC	XRA	H	EC	CPE	B ₁ B ₂
2D	DCR	L	6D	MOV	L, L	AD	XRA	L	ED	...	
2E	MVI	L, B ₂	6E	MOV	L, M	AE	XRA	M	EE	XRI	B ₂
2F	CMA		6F	MOV	L, A	AF	XRA	A	EF	RST	5
30	SIM		70	MOV	M, B	B0	ORA	B	F0	RP	
31	LXI	SP, B ₁ B ₂	71	MOV	M, C	B1	ORA	C	F1	POP	PSW
32	STA	B ₁ B ₂	72	MOV	M, D	B2	ORA	D	F2	JP	B ₁ B ₂
33	INX	SP	73	MOV	M, E	B3	ORA	E	F3	DI	
34	INR	M	74	MOV	M, H	B4	ORA	H	F4	CP	B ₁ B ₂
35	DCR	M	75	MOV	M, L	B5	ORA	L	F5	PUSH	PSW
36	MVI	M, B ₂	76	HLT		B6	ORA	M	F6	ORI	B ₂
37	STC		77	MOV	M, A	B7	ORA	A	F7	RST	6
38	...		78	MOV	A, B	B8	CMP	B	F8	RM	
39	DAD	SP	79	MOV	A, C	B9	CMP	C	F9	SPHL	
3A	LDA	B ₁ B ₂	7A	MOV	A, D	BA	CMP	D	FA	JM	B ₁ B ₂
3B	DCX	SP	7B	MOV	A, E	BB	CMP	E	FB	EI	
3C	INR	A	7C	MOV	A, H	BC	CMP	H	FC	CM	B ₁ B ₂
3D	DCR	A	7D	MOV	A, L	BD	CMP	L	FD	...	
3E	MVI	A, B ₂	7E	MOV	A, M	BE	CMP	M	FE	CPI	B ₂
3F	CMC		7F	MOV	A, A	BF	CMP	A	FF	RST	7

μCOM-85 ニーモニック ↔ 機械語 対照表

(MOV, INR, DCR, MVI) 命令

$r_2(r_1)$	A	B	C	D	E	H	L	M
MOV A, r_2	7F	78	79	7A	7B	7C	7D	7E
MOV B, r_2	47	40	41	42	43	44	45	46
MOV C, r_2	4F	48	49	4A	4B	4C	4D	4E
MOV D, r_2	57	50	51	52	53	54	55	56
MOV E, r_2	5F	58	59	5A	5B	5C	5D	5E
MOV H, r_2	67	60	61	62	63	64	65	66
MOV L, r_2	6F	68	69	6A	6B	6C	6D	6E
MOV M, r	77	70	71	72	73	74	75	—
INR r	3C	04	0C	14	1C	24	2C	34
DCR r	3D	05	0D	15	1D	25	2D	35
MVI r, B_2	3E	06	0E	16	1E	26	2E	36

(LXI, DAD, INX, DCX) 命令

\times	B	D	H	SP
LXI \times, B_2B_1	01	11	21	31
DAD \times	09	19	29	39
INX \times	03	13	23	33
DCX \times	0B	1B	2B	3B

スタック操作命令

\times	B	D	H	PSW
PUSH \times	C5	D5	E5	F5
POP \times	C1	D1	E1	F1

演算・論理 (I)

r	A	B	C	D	E	H	L	M
ADD r	87	80	81	82	83	84	85	86
ADC r	8F	88	89	8A	8B	8C	8D	8E
SUB r	97	90	91	92	93	94	95	96
SBB r	9F	98	99	9A	9B	9C	9D	9E
ANA r	A7	A0	A1	A2	A3	A4	A5	A6
XRA r	AF	A8	A9	AA	AB	AC	AD	AE
ORA r	B7	B0	B1	B2	B3	B4	B5	B6
CMP r	BF	B8	B9	BA	BB	BC	BD	BE

演算・論理 (II)

ADI B_2	C6
ACI B_2	CE
SUI B_2	D6
SBI B_2	DE
ANI B_2	E6
XRI B_2	EE
ORI B_2	F6
CPI B_2	FE

Accとのデータ転送(8ビット)

	STAX B	LDAX B	STAX D	LDAX D	STA B_2B_1	LDA B_2B_1
オペレーション	[(B)(C)] → (A)		[(D)(E)] → (A)		[(B ₂)(B ₁)] → (A)	
機械語	02	0A	12	1A	32	3A

回転命令

	RLC	RRC	RAL	RAR
オペレーション	$A_6 \rightarrow A_7, A_7 \rightarrow A_6$		$A_6 \rightarrow C, C \rightarrow A_7$	
機械語	07	0F	17	1F

HLレジスタとのデータ転送(16ビット)

	SHLD B_2B_1	LHLD B_2B_1	XTHL	XCHG	PCHL	SPHL
オペレーション	[(B ₂)(B ₁)] + 1, [(B ₂)(B ₁)] + 1		[(SP) + 1]	(D)(E)	(PC)	(SP)
機械語	22	2A	E3	EB	E9	F9

入出力・割込制御命令

	OUT B_2	IN B_2	D1	E1
機械語	D3	DB	F3	FB

Acc補正・C操作命令

	DAA	CMA	STC	CMC
オペレーション	補正	$A \leftarrow \bar{A}$	$C \leftarrow 1$	$C \leftarrow \bar{C}$
機械語	27	2F	37	3F

ジャンプ、コール、リターン命令

	→	NZ	Z	NC	C	PO	PE	P	M
JMP B_2B_1	C3	C2	CA	D2	DA	E2	EA	F2	FA
CALL B_2B_1	CD	C4	CC	D4	DC	E4	EC	F4	FC
RET	C9	C0	C8	D0	D8	E0	E8	F0	F8
条件	→	Z=0, Z=1	C=0, C=1	P=0, P=1	S=0, S=1				

その他の命令

	HLT	NOP
機械語	76	00

リスタート命令

	0	1	2	3	4	5	6	7
RST	C7	CF	D7	DF	E7	EF	F7	FF

85で追加された命令

	RIM	SIM
機械語	20	30



Something is Happening...

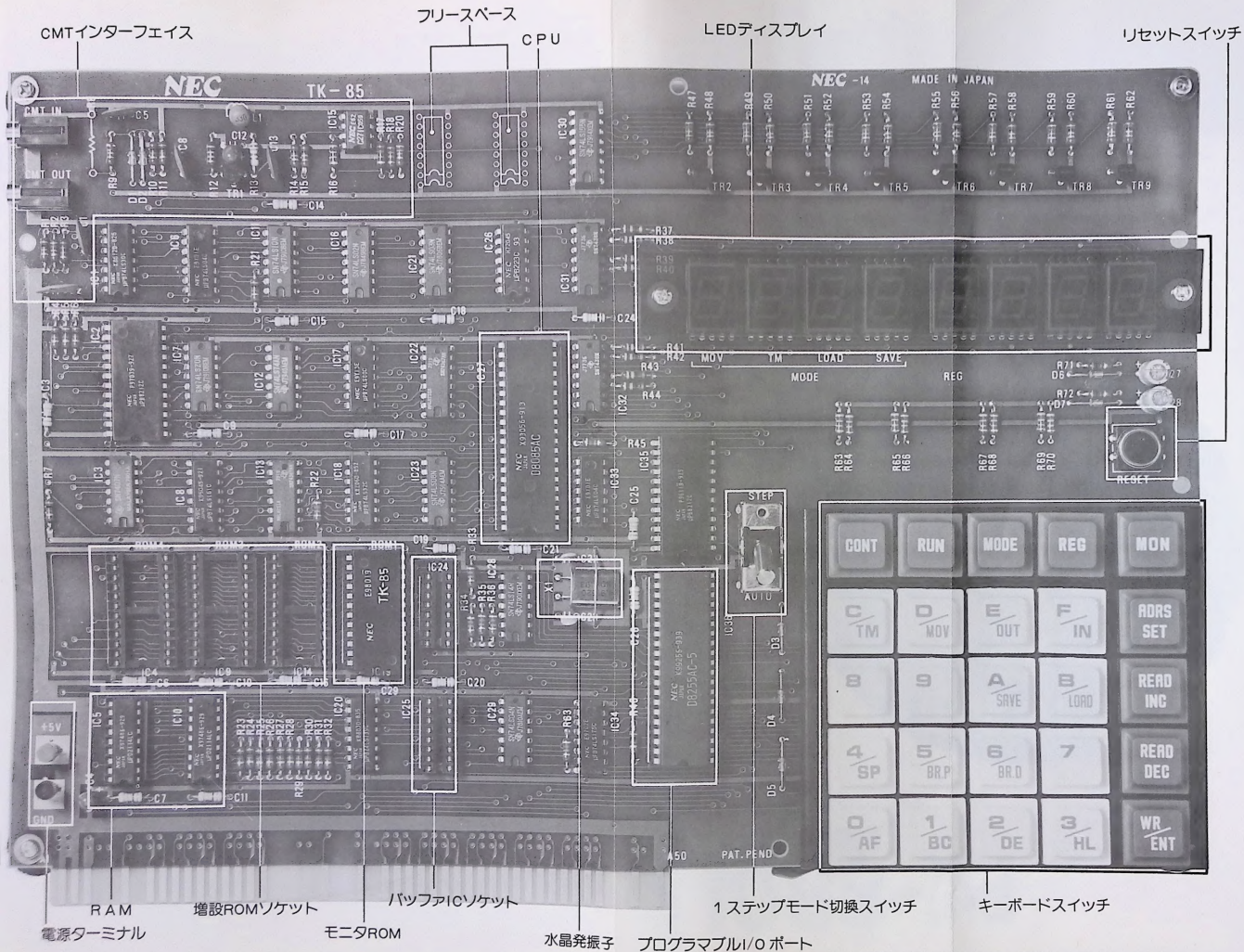
CMTインターフェイス

フリースペース

CPU

LEDディスプレイ

リセットスイッチ



〔本写真の使用部品は改良のため予告なく変更されることがあります〕